Actor Based Behavioural Simulation as an Aid for Organisational Decision Making

Souvik Barat

Director of Studies : Professor Balbir Barn

Supervisors : Professor Tony Clark

Vinay Kulkarni

Department of Computer Science

Middlesex University London

A thesis submitted to Middlesex University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

January 2019

Candidate's Declaration Form

Name of candidate : **Souvik Barat**

Student number : **M00516635**

Thesis Title : Actor Based Behavioural Simulation as an Aid for Organisational Decision Making

Degree for which thesis is submitted: Doctor of Philosophy

1. Statement of associated studies undertaken in connection with the programme of research (Regulation G3.1 refers)

While a registered student of the University, I attended the following courses /workshops / conferences:

- 11th Innovations in Software Engineering Conference (ISEC), February 9-11 2018, Hyderabad, India
- 50th Winter Simulation Conference (WSC), December 3-6 2017, Las Vegas, USA
- 31st Annual European Simulation and Modelling Conference (ESM), October 25-27 2017, Lisbon, Portugal
- 29th European Modeling and Simulation Symposium, September (EMSS), September 18-20 2017, Barcelona, Spain
- 15th International Conference on Practical Applications of Agents and Multi-Agent Systems, June 21 – 23 2017, Porto, Portugal (Paper Presentation and Doctoral Consortium)
- 10th Innovations in Software Engineering Conference (ISEC), February 5-7 2017, Jaipur, India

- 9th IFIP Working Conference on The Practice of Enterprise Modeling (PoEM), November 8-10 2016, Scovde, Sweden (Paper Presentation and Doctoral Consortium)
- 11th International Conference on Software Engineering and Applications (ICSOFT – EA), July 24-26 2016, Lisbon, Portugal
- 9th India Software Engineering Conference (ISEC), February 18-20 2016, Goa, India
- 18th International Conference on Model Driven Engineering Languages and Systems (MoDELS), September 28 – October 2 2015, Ottawa, Canada
- 47th Conference on Summer Computer Simulation (SCSC), July 26-29 2015, Chicago, USA

2. Concurrent registration for two or more academic awards

I declare that while registered as a candidate for the University's research degree, I have not been a registered candidate or an enrolled student for an award of another university, academic or professional institution.

3. Material submitted for another award

I declare that no material contained in the thesis has been used in any other submission for an academic award.

Signature of candidate: Souvik Barat Date : January 08, 2019

Acknowledgements

I would like to express my sincere gratitude to my supervisors Prof. Balbir Barn, Prof. Tony Clark and Vinay Kulkarni for their invaluable support and motivation. It would never have been possible for me to take this work to completion without their guidance and encouragement. I have greatly benefited from Prof. Balbir Barn for his thought provoking discussions and constructive feedback. His advice and suggestions on research methodologies, systematic study, creative thinking and research in general have helped in generating new thoughts and carrying out my research systematically. Thanks to Prof. Tony Clark for his scientific advice, sharing his technical knowledge and insightful discussions about research. His encouragement, patience and care have been invaluable. I cannot forget the help and valuable support that Vinay has extended to me. He has inspired me to focus on this research, provided all kinds of technical and non-technical support, and helped me to stay motivated throughout this journey. I have been extremely lucky to have such supervisors.

I would like to express my special thanks to my organisation, Tata Consultancy Services Limited, for allowing me to pursue this research and sponsoring my study.

I am truly grateful to my parents and brother for their encouragement and support. They helped me in all possible ways to reach this stage in my life. I would like to thank my wife Rupa for her inspiration, support and motivation to explore my potential and pursue my dreams. Thanks to my kids Arka and Ayan for their patience and understanding while I was busy with this research.

Finally, I wish to thank all my friends, relatives and colleagues of Tata Consultancy Services who have extended their valuable support during the course of this research.

Abstract

Decision-making is a critical activity for most of the modern organizations to stay competitive in rapidly changing business environment. Effective organisational decision-making requires deep understanding of various organisational aspects such as its goals, structure, business-as-usual operational processes, environment where it operates, and inherent characteristics of the change drivers that may impact the organisation. The size of a modern organisation, its socio-technical characteristics, inherent uncertainty, volatile operating environment, and prohibitively high cost of the incorrect decisions make decision-making a challenging endeavor.

While the enterprise modelling and simulation technologies have evolved into a mature discipline for understanding a range of engineering, defense and control systems, their application in organisational decision-making is considerably low. Current organisational decision-making approaches that are prevalent in practice are largely qualitative. Moreover, they mostly rely on human experts who are often aided with the primitive technologies such as spreadsheets and visual diagrams.

This thesis argues that the existing modelling and simulation technologies are neither suitable to represent organisation and decision artifacts in a comprehensive and machine-interpretable form nor do they comprehensively address the analysis needs. An approach that advances the modelling abstraction and analysis machinery for organisational decision-making is proposed. In particular, this thesis proposes a domain specific language to represent relevant aspects of an organisation for decision-making, establishes the relevance of a bottom-up simulation technique as a means for analysis, and introduces a method to utilise the proposed modelling abstraction, analysis technique, and analysis machinery in an effective and convenient manner.

Contents

Publications

Research Publications

2018

RP [1] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2018b). A Model Based Approach for Complex Dynamic Decision-Making. In *Communications in Computer and Information Science*, volume 880, pages 94–118. Springer

RP [2] Barat, S., Kulkarni, V., and Barn, B. (2018a). Towards Improved Organisational Decision-Making–A Method and Tool-chain. *Enterprise Modelling and Information Systems Architectures*, 13:6–31

2017

RP [3] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2017d). An actor-model based bottom-up simulation—An experiment on Indian demonetisation initiative. In *Winter Simulation Conference (WSC), 2017*, pages 860–871. IEEE

RP [4] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2017a). A domain specific language for complex dynamic decision making. In *ESM 2017: 31st Annual European Simulation and Modelling Conference*, pages 135–142 **(Best Paper)**

RP [5] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2017b). A method for effective use of enterprise modelling techniques in complex dynamic decision making. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 319–330. Springer

RP [6] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2017c). A Model based Realisation of Actor Model to Conceptualise an Aid for Complex Dynamic Decision-making. In *MODELSWARD 2017*, pages 605–616

RP [7] Barat, S., Rajbhoj, A., Kumar, P., and Kulkarni, V. (2017e). A Case Study Exploring Suitability of Bottom Up Modelling and Actor-based Simulation for Decision Making. In *Proceedings of Modeling Symposium*, pages 1–6

RP [8] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2015b). Supporting organisational decision making in presence of uncertainty. In *European Modeling and Simulation Symposium, EMSS 2017*, pages 87–101 **(Best Paper)**

2016

RP [9] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2016a). A conceptual model for organisational decision-making and its possible Realisations. In *ESM 2016: 30th Annual European Simulation and Modelling Conference*, pages 174–176

RP [10] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2016c). Enterprise modeling as a decision making aid: A systematic mapping study. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 289–298. Springer

RP [11] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2016b). A simulation-based aid for organisational decision-making. In *ICSOFT-EA 2016: 11th International Conference on Software Engineering and Applications*, pages 109–116

2015

RP [12] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2015c). Toward overcoming accidental complexity in organisational decision-making. In *Model Driven Engineering Languages and Systems (MODELS)*, pages 368–377

Doctoral Consortium

DC [1] Barat, S. (2017). An actor-based bottom-up Simulation aid for complex dynamic decision making. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 275–278. Springer

DC [2] Barat, S. (2016). A simulation based aid for complex dynamic decision making. In *PoEM Doctoral Consortium*, pages 22–31

Publications from Overarching Research Initiative

2018

OP [1] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2018). A homogeneous actor-based monitor language for adaptive behaviour. In *Programming with Actors*, pages 216–244. Springer

2017

OP [2] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2017c). ESL: An Actor-Based Platform for Developing Emergent Behaviour Organisation Simulations. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 311–315. Springer

OP [3] Clark, T., Barn, B., Kulkarni, V., and Barat, S. (2017a). Querying histories of organisation simulations. *Information Systems Development (ISD) - Advances in Methods, Tools and Management*, 9:1–12

OP [4] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2017b). Actor monitors for adaptive behaviour. In *Proceedings of the 10th Innovations in Software Engineering Conference*, pages 85–95. ACM

OP [5] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2017d). Generating Filmstrip Models from Actor-Based Systems. In *MODELS (Satellite Events) 2017*, pages 576–582

OP [6] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2017e). The Construction and Interrogation of Actor Based Simulation Histories. In *ER Forum/Demos 2017*, pages 320–333

2015

OP [7] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2015d). Using simulation to address intrinsic complexity in multi-modelling of enterprises for decision making. In *Proceedings of the Conference on Summer Computer Simulation*, pages 1–11. Society for Computer Simulation International

OP [8] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2015a). A wide-spectrum approach to modelling and analysis of organisation for machine-assisted decision-making. In *Workshop on Enterprise and Organizational Modeling and Simulation*, pages 87–101

2014

OP [9] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2014). Model Based Enterprise Simulation and Analysis - A Pragmatic Approach Reducing the Burden on Experts. In *ER Workshops 2014*, pages 3–12

OP [10] Clark, T., Kulkarni, V., Barn, B., France, R., Frank, U., and Turk, D. (2014). Towards the model driven organization. In *47th Hawaii International Conference on System Sciences (HICSS)*, pages 4817–4826. IEEE

List of Figures

List of Tables

List of Abbreviations

4EM	For Enterprise Modeling
ABCL	Actor-Based Concurrent Language
AI	Artificial Intelligence
ARIS	ARchitecture of Integrated Information Systems
AST	Abstract Syntax Tree
BAU	business as usual
BDI	Belief-Desire-Intention
BMM	Business Motivation Model
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPO	Business Process Outsourcing
CAS	Complex Adaptive Systems
CIMOSA	Computer Integrated Manufacturing Open Systems Architecture Framework
CTL	Computation Tree Logic
DEMO	Design and Engineering Methodology for Organizations
DESIRE	DEsign Specification of Interacting REasoning components
DEVS	Discrete Event System Specification

DoDAF	Department of Defense Architecture Framework
DSL	Domain Specific Language
DSR	Design Science Research
DSRM	Design Science Research Methodology
EA	Enterprise Architecture
EBNF	Extended Backus-Naur-Form
EEML	Extended Enterprise Modelling Language
EIF	European Interoperability Framework
EKD	Enterprise Knowledge Development
EM	Enterprise Modelling
EPC	Event-driven Process Chain
ESL	Enterprise Simulation Language
FTE	full time employee
GAML	GAma Modeling Language
GERAM	Generalized Enterprise Reference Architecture and Methodology
GIM	GRAI Integrated Methodology
GRAI	Graphs with Results and Actions Interrelated
GST	General System Theory
IDE	integrated development environment
IDEF	Integration DEFinition
IEM	Integrated Enterprise Modeling
IS	Information System
IT	Information Technology
JADE	Java Agent DEvelopment Framework
JVM	Java Virtual Machine

KAOS	Knowledge Acquisition in automated specification or Keep All Objectives Satisfied
LTL	Linear Temporal Logic
MDA	Model Driven Architecture
MDO	Model Driven Organisation
MEMO	Multi-Perspective Enterprise Modelling
MoDAF	British Ministry of Defence Architecture Framework
NSS	National Students Survey
OMG	Object Modeling Group
OR	Operational Research
PERA	Purdue Enterprise Reference Framework
PIM	Platform Independent Model
PIM	Platform Specific Model
POJO	Plain Old Java Object
RFP	Request for Proposal
RM-ODP	Reference Model of Open Distributed Processing
SD	System Dynamics
SEAM	Systemic Enterprise Architecture Methodology
SLR	Systematic Literature Review
SMS	Systematic Mapping Study
SnF	Stock and Flow
SSPO	Software Service Provisioning Organisation
SVBR	Semantics of Business Vocabulary and Rules

ToGAF The Open Group Architecture Framework

UEML Unifed Enterprise Modeling Language

UK United Kingdom

UML Unified Modelling Language

Font Style Convention

The font style convention followed in this thesis are:

Words in italics	:	Concepts, words to introduce new term, emphasis of words, and variables
`Words in teletype font`	:	Meta model elements, code fragments
'*Words in italics with single quotes*'	:	Model instances

Chapter 1

Introduction

1.1 Research overview

Modern organisations repeatedly evaluate their status-quo and make decisions to stay competitive and *economically viable* [185] in a dynamic business environment that experiences globalisation, intense competition, and technology innovations [70]. In this endeavour, the decision-makers of the organisations constantly explore the answers for a range of decision questions such as: Is the current state and form of the organisation appropriate to stay ahead of the competition or economically viable? If not, *What* kind of changes are necessary to achieve organisational goals? *When* to apply those changes? *Where* to apply those change? and *How*?

Predicting precise answers to these decision questions is extremely important for organisations as an inaccurate answer may lead to an ineffective decision and the cost of such a decision is often prohibitively high in reality. Moreover an inappropriate decision limits the possibility of suitable adaptation options later [177]. Therefore, decision makers are additionally tasked to anticipate the consequences that include the evaluation of the *utility*, short-term and long-term *implication*, and *risk* of a decision prior to its implementation.

Deciding on an effective decision with best possible *consequences* requires precise analysis of various aspects of an organisation, such as goals, organisational structure, operational processes, historical data, and its operating environment [177]. The inherent characteristics of the modern organisation that include its socio-technical characteristics [132], complex and dynamic organisational structure [141], nonlinearities in the interactions with its environment [126, 62], unaccounted delays [185], inherent uncertainty [62] and emergent behaviour [150] make decision-making *exceedingly complex* [183].

The state-of-the-practice of organisational decision-making chiefly relies on the *qualitative* approaches [152], such as discussion and interviews, with a minimum *quantitative* assistance that comes from spreadsheets based data computation and analyses [126]. This excessive dependency on human intuitions and interpretations compounded with inadequate quantitative analysis often results in a less effective decision especially when the context is complex and dynamic [135, 105]. A suitable combination of *qualitative* and *quantitative* approaches as suggested by Kaplan in [106] is needed in this context.

A range of enterprise modelling and analysis techniques supporting quantitative approaches exist. However, their utility is limited to a class of decision-making as compared to a wide range discussed in management literature [68, 8, 141, 66, 177]. For example, *inferential techniques* [138] that rely on the statistical and mathematical interpretation of historical system data are suitable only for static environments (*i.e.*, the environmental and organisational topologies are fairly static with the time); mathematic models, such as linear programming [48] and integer programming [176], work well for mechanistic systems that are not characterised as autonomous, adaptable and uncertain; the enterprise models, such as ArchiMate[100], i* [218], and Business Process Model and Notation (BPMN) [209], are found to be inappropriate for the systems that exhibit uncertainty and emergent behaviour; whereas the actor and agent based models that are based on the *actor model of computation* [2, 96] and agent-based systems [129] fall short for expressing the complex organisational structure and uncertainty.

This research aims to supplement the state-of-the-practice of organisational decision-making with appropriate technology aids to comprehend the necessary aspects of the organisation, explore decision space, understand the consequences of possible decision alternatives, and produce sufficient quantitative evidence to arrive at effective decisions. The necessary aspects and characteristics of complex organisations for an effective decision-making are ascertained by reflecting on organisational theory [206, 14, 10, 7] and management literature [180, 182]. The research contributions are: a domain specific modelling language to capture necessary aspects and their characteristics, an approach to analyse decision alternatives and understand their consequences, and a method to capture organisational aspects and perform various *what-if* analyses leading to an effective decision-making.

The proposed modeling language considers six interrogative aspects: *why*, *what*, *how*, *when*, *where*, and *who*, as suggested by Zachman in [220] as the necessary aspects for organisational decision-making. The proposed modelling approach primarily differs from conventional En-

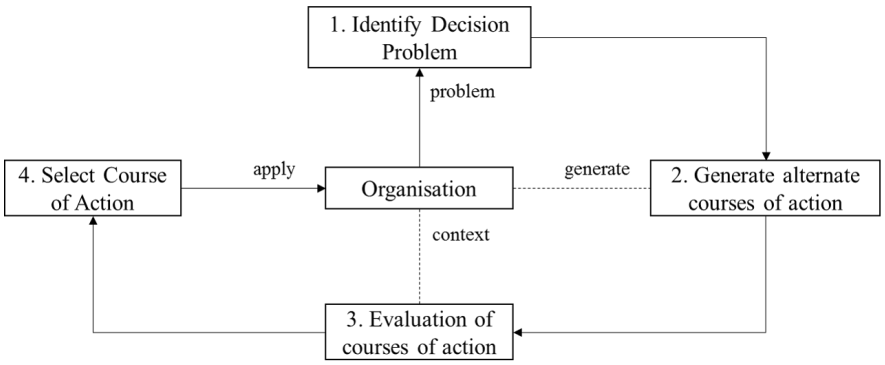


Figure 1.1 Organisational decision-making

terprise Modelling (EM) as it considers an organisation as *system of systems* as opposed to monolithic unit. The proposed approach extends the notion of *actor* [2, 96] to capture all necessary aspects (*i.e.* six interrogative aspects proposed by Zachman) and the desired characteristics of a complex organisation, such as autonomy, adaptability and uncertainty. The proposed analysis approach draws upon bottom-up [192] simulation technique to understand the emergent behaviour of the organisation [150], which is critical to understand modern organisations. The proposed method is constructed based on the best practice of simulation based analysis [174] and management view of decision-making [70]. In brief, the research proposes an approach to represent a view of a real organisation using a purposive model, and perform the *what-if* analyses by hypothesising possible changes on the constructed model and iterating the simulation on modified model. It assumes that multiple iterations exploring the possible changes and a human-in-the-loop comparative analysis of the simulation outcomes can lead to an effective quantitative approach.

This chapter briefly introduces the research presented in this thesis by describing the precise problem statement, objectives and aims, research questions, hypotheses, an overview of the proposed approach, and it concludes with a chapter outline of this thesis.

1.2 Problem statement and research objectives

Organisational decision-making is a process of selecting suitable course of action that has potential to achieve the organizational objectives or goals [130]. Kickert et al. define decision-making as a problem solving method that aims to improve a certain organisational or managerial status-quo in a desired direction [108]. Management theories [141, 8, 130, 67, 75] consider organisational decision-making as intentional and consequential action where the most effective

solution is chosen from a list of alternatives by adopting an appropriate viewpoint that ranges from a rational model [8] to an anarchic model [60]. Methodologically, the management literature approaches decision-making using four major steps: (1) problem identification, (2) generation of alternative courses of action, (3) evaluation of courses of action, and (4) selection of the most effective or feasible course of action as depicted in Figure 1.1.

The state-of-the-practice of organisational decision-making reflects on the intuitions and cognitive abilities of the decision makers, and adopts interpretive approaches to perform the decision-making steps noted above. The state-of-the-art modelling and analysis approaches are best suited to perform step (3) of the repetitive *programmed* decision-making [67]. They fail to demonstrate the expected efficacy for *nonprogrammed* decision-making [67] that is non repetitive in nature and characterised by inherent uncertainties and significant ambiguities. This research conceptualises and develops an effective technological aid to perform the organisational decision-making steps depicted in Figure 1.1 for *programmed* and *nonprogrammed* decision-making. The specific objectives are:

1. Capture decision problem and the context of the organisational decision-making in a precise and unambiguous form, *i.e.*, a modelling aid for step 1 .

2. Capture alternative courses of action, *i.e.*, a modelling aid for step 2.

3. Understand the short-term and long-term *consequences* of all possible alternatives a-priori and using quantitative terms, *i.e.*, a quantitative analysis aid for step 3.

It is argued in this thesis that the adequate modelling and quantitative analyses support to perform steps 1, step 2 and step 3 of Figure 1.1 help an effective organisation decision-making in step 4.

This research is conceptualised as a specialised stream of an overarching research initiative, Model Driven Organisation (MDO)[1], which aims to promote models and modelling capabilities to address strategic, tactical and operational needs of the modern enterprises as highlighted in [OR11] (*see* the list of *Publications from Overarching Research Initiative*).

The overarching MDO research initiative presents a general purpose actor-based simulation language termed as Enterprise Simulation Language (ESL) [OP2, OP3], ESL simulation engine [OP2], and a set of supporting technologies such the monitor technology [OP1, OP5], query on actor history [OP4, OP7], and visaulisation of simulation history using filmstrip [OP7]. The

[1] http://sites.tcs.com/innovation-forum/model-driven-organization

research presented in this thesis specialises the focus to the modelling, analysis techniques and technologies to address organisational decision-making.

1.3 Research questions

Information System (IS) research recommends that information should be amenable for computational analysis and technology-aided analysis for better sense-making, understanding and prediction [200, 219, 134]. In congruence to this viewpoint, this research explores the possibility of utilising the technology-aided analysis to perform the decision-making steps depicted in Figure 1.1. This research focuses on three broad research questions as follows:

> [RQ1] What information of an organisation and its environment are necessary to understand the possible consequences of prospective courses of action in an organisational decision-making?
>
> [RQ2] What kinds of modelling capabilities are expected to capture a decision problem, possible courses of action, and necessary information of an organisational decision-making in a precise and machine-interpretable form?
>
> [RQ3] What kinds of analysis technique are appropriate to analyse the captured information and produce necessary quantitative evidences for effective organisational decision-making?

This research also focuses on an effective methodological rigour to identify the relevant information, capture them using machine interpretable form, and perform required analyses in a way that conforms to the management viewpoints.

This research primarily limits the exploration to the management and IS research. The political [157], ethical [197], psychological, and the power [215] considerations of organisational decision-making are considered as out of scope of this research.

1.4 Hypotheses

As suggested in the Zachman framework [220], this research makes an assumption that the *why*, *what*, *how*, *when*, *where*, and *who* are necessary and sufficient information for organisational decision-making. This research further hypothesises that the *actor model of computation* [2] is

an effective modelling abstraction to capture complex dynamic organisation, and a bottom-up behavioural simulation technique is an effective means to analyse alternative courses of action and understand their consequences a-priori. Specific research contributions conforming to the proposed hypotheses are described in the next section.

1.5 Research method, contributions and validation

A Design Science Research (DSR) philosophy in line with the guidelines proposed by Hevner et al. [95] is adopted in this research to understand the problem space, conceptualise a pragmatic approach, develop research contributions and validate their efficacies.

The research starts with an exploration of the management literature that focuses on the *organizational theory* [68, 66, 10, 191], *decision making* [62, 177, 185], management theories on the *decision making methods* [8, 66, 141, 60] to understand the problem space. Then, it evaluates the state-of-the-art modelling and analysis techniques, such as Enterprise Modelling [100, 204, 218, 209], Actor technologies [90, 5, 15, 187, 198, 12], the simulation literature [31, 178, 129, 47, 174] and Model Driven Engineering, to understand the existing technological capabilities and ascertain their suitability in the context of organisational decision-making. The research shows the precise inadequacies of the state-of-the-art modelling and analysis capabilities, and finally it proposes a conceptual approach supported by a proof-of-concept technological aid implementation for quantitative analysis exploring decision alternatives.

A Domain Specific Language (DSL) to represent the necessary information for an organisational decision-making, a simulation-based analysis technique for *what-if* scenario playing and a method to use proposed research artifacts are presented as part of this research. The key research contributions are four-fold:

Contribution 1: Concepts of organisational decision-making: A conceptual model highlighting the relevant *concepts* and their relationships to sufficiently describe an organisational decision-making problem. The concepts are discerned from the management literature and expressed using a meta-model. From DSR standpoint, the proposed conceptual model is a *Constructs* artifact [94] of this research.

Contribution 2: Meta-model and a DSL: A meta-model called OrgML to capture necessary information for organisational decision-making. Its caters to relevant aspects of the organisation, its socio-technical characteristics, and associated uncertainties. This is a

Model artifact [94] in terms of DSR terminology. This research also presents a domain specific language, named as OrgML, that realises the proposed OrgML meta model. The OrgML can be seen as an *Instantiation* artifact from DSR perspective.

Contribution 3: An approach to convert captured information into simulatable form: An actor-based simulation language, named as ESL, is considered as an underlying simulation language for *what-if* analysis. This research proposes an approach to transform the captured information, *i.e.* OrgML specification, into an ESL specification using a model-to-model transformation technique. From DSR perspective, the proposed transformation schema can be visualised as a *Method* artifact and transformation program as an *Instantiation* artifact.

Contribution 4: Method: An integrated and iterative method to construct the purposive simulation model leading to the organisational decision-making in a systematic manner. The proposed method supports: (i) construction of a simulation model from available information of an organisation, (ii) model validity, and (iii) simulation of constructed model for *what-if* analyses. It extends the modelling and model validation methods advocated by Robert Sargent [174] and refines the management view of decision-making advocated by Richard Daft [70]. From DSR standpoint, the proposed method is a *Method* artifact.

This research claims that the proposed *concepts of organisational decision-making*, *i.e.* contribution 1, is a precise modelling and analysis requirement for an effective organisational decision-making. This has an applicability beyond this research as it can be considered as requirements to improve the state-of-the-practice of organisational decision-making. OrgML is an advance over existing enterprise modeling and actor languages and it is a DSL for organisational decision-making. The proposed approach to convert OrgML specification into simulatable form, *i.e.* contribution 3, and proposed method, *i.e.*, Contribution 4, are novel contributions from a methodology perspective as they show how technological aids can be systematically and meaningfully used in organisational decision-making.

The *efficacy* of these proposed research contributions, *i.e. Constructs*, *Model*, *Method* and their *Instantiations*, are validated through an *Artificial* and *Ex-Post* [161] validation strategy. Artificial yet close to real life case studies that are considered for research validation are described below:

Case Study 1: Software Service provisioning organisation: This case study models a realistic Software Service Provisioning Organisation (SSPO), whose goal is to improve the profit margin, maintain the quality of the developed software, and ensure on-time software delivery. The short-term and long term implication of various courses of action, such as the increase of resource strengths, improve the resource skills, and the use of effective software development tools, are evaluated using proposed approach.

Case Study 2: Business Process Outsourcing organisation: This case study considers a decision-making case from the Business Process Outsourcing (BPO) industry. The proposed approach is used to explore BPO improvement initiatives, such as competitive pricing model, in a competitive business environment. The simulation is used to understand the consequences of the courses of action over the time horizon.

Case Study 3: Demonetisation: This case study imitates a subset of the Indian Demonetisation initiative[2] and explores various actions that could have controlled the chaos emerged in the initial stage of the Demonetisation in an effective manner.

Case Study 4: University: This case study models and simulate a University, whose aim is to improve its ranking in terms of research index, teaching quality, and students' satisfaction index. The proposed OrgML is used to model a University and an ESL based simulation is used to select an effective course of action from a range of possibilities that include the change in academic student ratio, prioritise teaching and research focus of the academics, and adopt better timetable.

The research outcomes are communicated to the conference and journal papers. The problem statement is presented in [RP5, DC1, DC2]. The inadequacy of the existing modelling and analysis techniques is presented in [RP8, RP10]. The research contributions are presented in [RP1, RP3, RP4, RP5, RP7]. The validation of the research contributions is reported in multiple publications: Software Service Provisioning Organisation is illustrated in [RP3, RP6, RP9], the case study on Business Process Outsourcing organisation is presented in [RP5], and the Demonetisation case study is discussed in [RP2].

The rest of this thesis uses a subset of the University case study as a running example to illustrate the proposed concepts and contributions. A brief overview of the University case study is discussed in the next section.

[2]https://en.wikipedia.org/wiki/2016_Indian_banknote_demonetisation

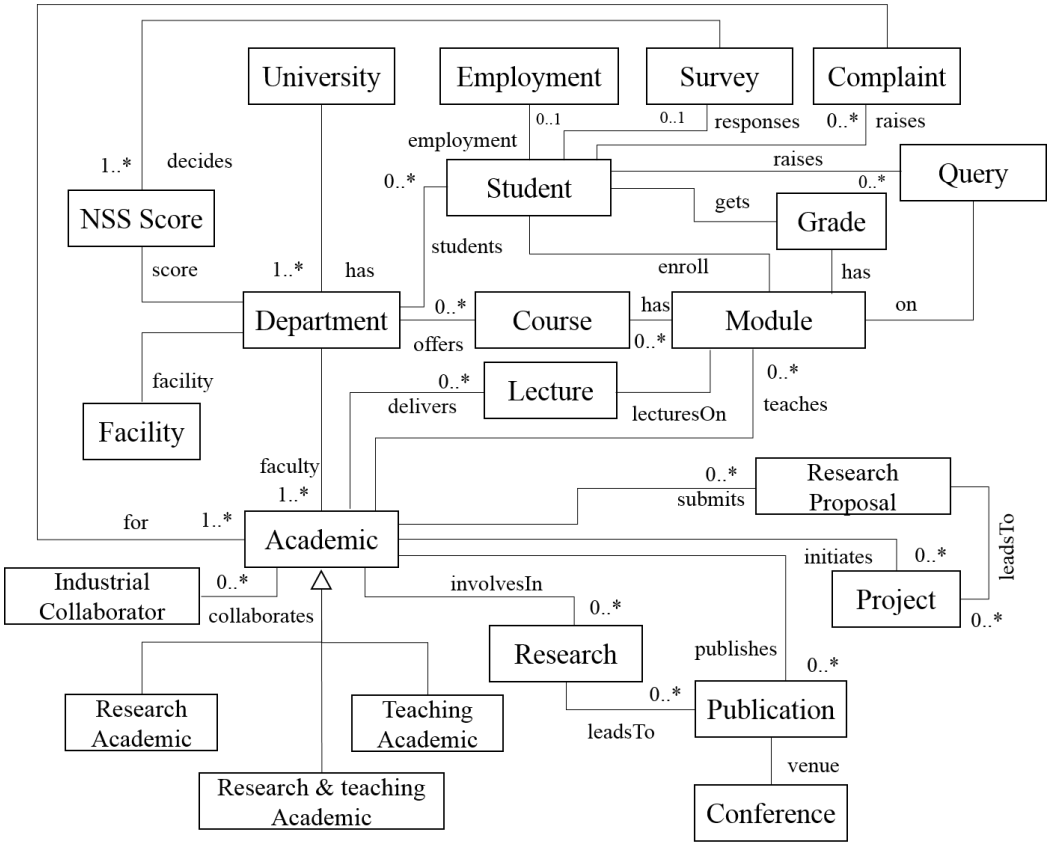


Figure 1.2 Structural description of University

1.6 An illustrative example

Considers a hypothetical university from United Kingdom (UK), lets refer to it as ABC University, aims to improve its ranking [6] along academic position and research ranking. In UK, the academic position is typically measured using the academic results of the final year students, the employability of outgoing students and the National Students Survey (NSS) score (the NSS score is an outcome of the students survey on various aspects that includes the quality and speed of feedback to the assessments, the employability options, quality of teaching and facilities). The research ranking is primarily measured based on the research outcomes such as the publication records at top-tier conferences and journals, research grants, and industry collaborations [128]. This section briefly introduces ABC University and discusses decision-making scenarios.

1.6.1 Description

The academics focus on teaching activities, such as offering course modules and student assessment. They also conduct research in various topics and publish papers at top-tier conferences

Table 1.1 Activities of Academics and Students

Active Element	Activities
Research Academic	Research, Paper Writing, Writing Research Grand Proposal, Project Work, Research Collaboration, Query Resolution, Complain Resolution
Teaching Academic	Prepare for Lecture, Delivering Lecture, Prepare for Student Assessment, Student Assessment, Project Work, Query Resolution, Complain Resolution
Research and Teaching Academic	The combination of the activities of Research Academic and Teaching Academic
Student	Attend Lecture, Self Study, Appear for Assessment, Raise Query, Raise Complaint, Response to NSS Survey

and journals. The key concepts of ABC university that are considered are illustrated using a class diagram in Figure 1.2. As shown in the figure, a `Department` is formed using a set of `Academics`; `Departments` offers a set of `Courses` where each `Course` have multiple `Modules`.

ABC university has three types of `Academics`: `Research academic`, `Teaching academic`, and `Research & Teaching academic`. The `Research` academics involve in research activities that include preparation and submission of `Research Grant Proposals`, conduct `Research` work, and publish `Papers` in quality `Conferences` and journals. The `Teaching` academics prepare and deliver `Lectures`, prepare student `Assessments`, evaluate `Students`, and publish `Grades`. The `Research and Teaching` academics are involved in both kinds of activities. In addition, all these three types of `Academics` work on `Projects` and `Industrial collaborations`, clarify student `Queries`, and resolve student `Complaints`. The `Students` attend `Lectures` for their enrolled `Modules`, appear for `Assessments`, and get `Grades`. The `Students` may raise `Queries` in case of any doubt in the `Lectures` and they may raise a `Complaint` for some unanswered concerns. The final year `Students` response to the `NSS Survey` that results into the `NSS Score` of their respective `Department` and `University`. Finally, the `Students` complete the `Course` with consolidated `Grade`, get `Degree`, and may get an `Employment` in an academic or industrial institution.

In this setting, the behaviour of `University` and `Departments` emerge from the behaviours and interactions of their constituent active elements such as `Academics` and `Students`. These constituent elements are capable of performing a set activities as listed in Table 1.1 and they choose to perform a specific activity (based on their state, objectives and interest) for a specific time slot. There can be an inherent uncertainty and nonlinearity. For example, they may randomly choose an activity at a given time and interrupt an activity at any time for a high priority work. Moreover, their earlier activities, such as inadequate preparation for a lecture

or incomplete research work, may force them to behave differently than the expected normal behaviour.

1.6.2 Decision space exploration scenarios

Influential world ranking tables, such as THE[3] and ARWU[4], publish indicative improvement factors[5] or potential courses of action [128]. The courses of action which are commonly discussed in this context are:

- Academic and student ratio.
- Balance between research and teaching academics.
- Work priorities of the academics.
- Appropriate timetabling.
- Experience and academic records of the academics.
- Industrial collaboration

However, it is not known which course of action is the most effective for a University as it depends on several internal and environmental factors such as current state of the university, its strengths, quality of potential students. The decision makers, such as a Dean and Head of Department, attempt to understand the complex behaviour and associated dynamism to explore the decision questions such: What alternative or set of alternatives are effective to improve the teaching and research indicators? How do the alternatives compare? Are there any negative consequences of a chosen alternative? When will the university will start observing the benefit (*i.e.* near-term or long-term) for change? This research demonstrates how these decision-making questions can be answered in a quantitative form and how they lead to decision-making.

1.7 Thesis structure

The rest of the thesis is structured as follows:

Chapter 2 : Research methodology - This chapter presents an overview of the methodological foundations and justifies the suitability of the DSR methodology to conduct research activities and validate research outcomes. A schema illustrating how the activities of this research are conducted using the DSR methodology is discussed in this chapter.

[3] https://www.timeshighereducation.com

[4] http://www.shanghairanking.com

[5] https://www.topuniversities.com/student-info/university-news/university-oxford-tops-times-higher-education-world-university-rankings-2018

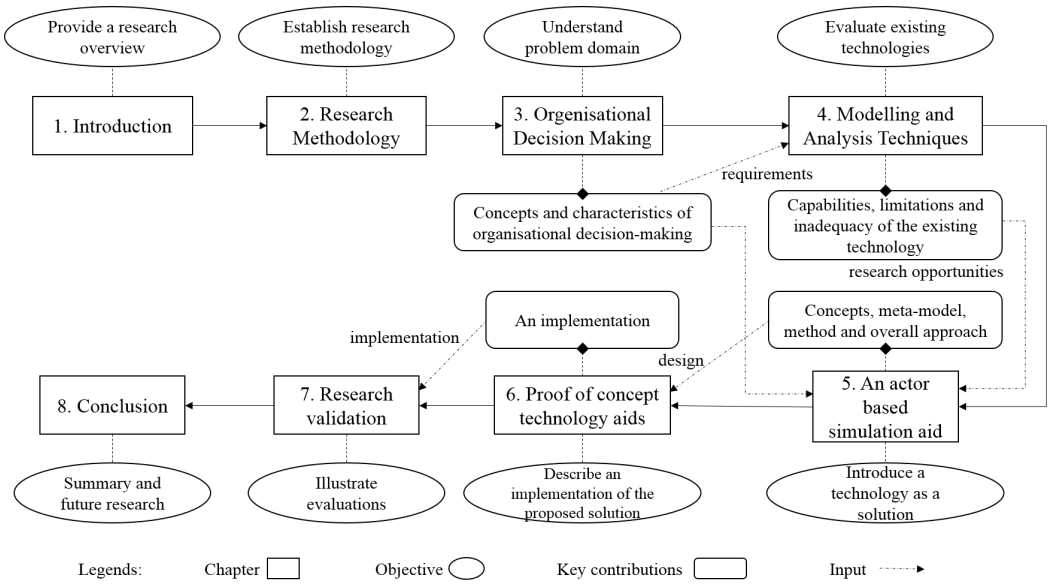


Figure 1.3 Thesis structure, objectives and contributions

Chapter 3 : Organisational decision-making: Explores the problem space by reviewing the management literature on organisational decision-making. It presents the theoretical and practical foundations of organisational decision-making, summarises the relevant concepts using a *concept* meta-model, and discerns the modelling and analysis needs for an effective organisational decision-making. Principally, this chapter focuses on the research question *QR1* and presents *Contribution 1* as a solution to RQ1.

Chapter 4 : Modelling and analysis techniques: Presents systematic literature reviews on a spectrum of modelling and analysis techniques, such as enterprise modelling and actor/agent technologies, to evaluate their capabilities with respect to the modelling and analysis requirements presented in Chapter 3. It also discusses specific inadequacies of the state-of-the-art modelling and analysis techniques in the context of organisational decision-making.

Chapter 5 : An actor based simulation aid: Presents the core contributions of this research. This chapter focuses on the research questions *RQ2* and *RQ3*, and presents the *Contribution 2* (*i.e.*, OrgML meta-model), *Contribution 3* (*i.e.*, OrgML to ESL transformation strategy), and *Contribution 4* (*i.e.*, method for model construction, validation and what-if analysis) to realise the research objective outlined in this chapter.

Chapter 6 : Proof of concept technology aids: Presents a proof-of-concept implementation of the proposed approach that include OrgML, OrgML to ESL translator, a tool to visualise

the key performance indicators of the organisation, and an overall simulation framework for organisational decision-making.

Chapter 7: Research validation: This chapter validates the research outcomes using three techniques - (i) artificial case-study based approach where four case studies are considered as discussed in section 1.5 (ii) comparative approach by considering popular modelling and simulation techniques such as System Dynamics (SD) model, and (iii) argumentative approach by considering the modelling and analysis needs discussed in Chapter 3. This chapter also discusses limitations of the proposed approach and scope for further improvements.

Chapter 8 : Conclusion: Concludes this thesis by highlighting the key contributions, potential improvements of the state-of-the-practice of organisational decision-making, and a future research agenda.

The overview of the thesis structure, key objectives of the each chapter, the research contributions of the specific chapters are illustrated in Figure 1.3.

1.8 Summary

This chapter highlights a brief overview of the research presented in this thesis. It includes a justification of the research objective to introduce technology-driven quantitative analysis for organisational decision-making as opposed to qualitative approaches. Three research questions that interrogate the necessary information, an intuitive representation, and suitable analysis technique for the organisational decision-making are discussed. An introductory overview of the thesis that adopts the *actor model of computation* as modelling abstraction and the simulation method for organisational decision-making are presented in this chapter.

Chapter 2

Research Methodology

Research is often initiated when there is a need to find a solution for a known problem (*i.e., Improvement*), extend a known solution to a new problem (*i.e., Adaptation*) or invent a new solution for new problem (*i.e., Invention*) [86].

This chapter sets out the philosophical grounding, methodology and a plan to conduct this research. Section 2.1 discusses philosophies and methodological approaches that are explored to establish a methodological viewpoint for this research. Section 2.2 briefly reviews the Design Science Research (DSR) paradigm as it is considered as a basis for conducting this research and validating research artifacts. The chapter concludes with a method with specific steps for conducting research in section 2.3.

2.1 Philosophical grounding

A research philosophy is the 'basic belief system or world view that guides the investigation' [89]. In IS, the belief system of the researchers is typically framed using four philosophical groundings: *ontology*, *epistemology*, *methodology* and *axiology* [149, 1]. The *ontology* is a set of key concepts that sufficiently describe the belief system of a reality or a domain of interest, the *epistemology* is how researchers think or a reflection of the reality, *methodology* is plan describing how researchers are going to use an existing knowledge-base or epistemology to gain new knowledge, and *axiology* is the morals or ethical considerations.

Researchers consider different philosophical standpoints as *research paradigm* to explore various research problems [149]. For example, a set of researchers may attempt to understand a system by quantitative analysis of empirical data or subjective analysis of multiple viewpoints

Table 2.1 Philosophical Assumptions (Source [1])

Research Paradigm	**Philosophical Assumptions**			
	Ontology	**Epistemology**	**Methodology**	**Axiology**
Positivist	Single, stable reality	Objective	Experimental, Quantitative, Hypothesis testing	Prediction
Interpre-tive	Multiple Realities	Subjective	Interactional, Interpretation, Qualitative	Contextual Understanding
Critical	Socially Constructed	Social and Political	Discourse Analysis	Contextual Analysis considering researchers' value system
Design Science	Contextually situated realities	Iterative	Developmental and *Knowing through making*	Improvement through control, creation and progress

of a system. The system under consideration, existing knowledge about the system and veracity of the knowledge help decide appropriate research paradigm. The notable research paradigms in IS research are: *positivist* [145], *interpretive* [85], *critical research* [152] and *design science research* [95]. The philosophical standpoint of four research paradigms, as discussed in [1], are described in Table 2.1.

As described in the table, *positivist research* tries to gain knowledge through observations of real system. It uses the empirical methods such as *measurement* and *hypothesis testing*. The positivist research is a useful paradigm where a single truth about the reality or sufficient data about the truth exists. An *Interpretive research* attempts to make sense of a system through subjective evaluation from the perception of key stakeholders. It adopts qualitative methods such as *case studies*, *interviews*, *observations* and *action research* to develop knowledge. Interpretive paradigm is typically considered in a situation where getting sufficient data from reality is hard but knowledgeable people exist. Similar to interpretive research, the *Critical research* considers subjective evaluation with an additional focus on ethical consideration. The qualitative methods such as *ethnography* and *action research* focusing on political, cultural and power relations in a social setting are the basis for developing knowledge in critical research. Socio-economical imbalance due to inadequate or inappropriate policies are subject for critical research.

The DSR relies on *science of the artificial* [184]. It focuses on man-made artifacts, such as a model, as the basis to gain the knowledge as opposed to interpreting real system or system related data. The DSR approach is useful when there is no single truth about the system so

Table 2.2 Design-Science Research Guidelines (Source [95])

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to address problems, which important and relevant for business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Contributions	Effective design-science research must provide clear and verifiable contributions.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires innovative utilisation of the available means without violating the laws exist in the problem space.
Guideline 7: Communication of Research	Design-science research must be presented effectively to technology-oriented as well as management-oriented audiences.

as to adopt positivist research, considering interpretive research is infeasible or they demand significant effort and time.

The key activities of this research include: (i) understand problem space (*i.e.* organisational decision-making) and existing solutions, (ii) conceptualise a new solution to *improve* state-of-the-practice of organisational decision-making and state-of-the-art of modelling and analysis techniques, and (iii) validate the *efficacy* of the proposed solution. Understanding problem space by observing real organisational decision-making and validating the proposed solution in real organisational setting are time and cost intensive activities. Moreover, there is no universally accepted quantitative means that can establish the '*efficacy*' of an organisational decision-making solution. Thus the positivist approach is not an effective proposition for this research. Similarly the interpretive approach and critical research are difficult to adopt as the availability multiple key decision makers to express their subjective views is a concern. The Design Science Research (DSR) methodology [94] is less dependent on the real system and/or key stakeholders. It relies on an incremental approach to develop research artifacts and a synthetic environment for research validation. Thus, this research draws an ontological and epistemological foundation from the fields of organisational theory [66, 68, 126] and organisational decision-making [62, 141, 185, 177]; it adopts Design Science Research (DSR) [94] as methodological foundation for conducting research activities and validating research artifacts; and considers incremental improvement as an axiological assumption. The next section

presents the core concepts of DSR that help formalise a plan to conduct research activities in a precise and systematic manner.

2.2 Design Science Research

The Design Science Research (DSR) methodology reflects on artificial artifacts, such as design, reflection and abstraction, for problem solving and knowledge creation. The key considerations of DSR methodology are: how to construct an artificial representation, how to establish the truthfulness of the artificial representation, how to analyse an artificial representation for gaining knowledge about a real system and how to correlate analysis results with the reality. The principal guidelines of DSR paradigm (presented in [95]) are described in Table 2.2. As described in the table, it considers *design* as the core research contribution, recommends a methodology for conducting research, proposes validation strategies to evaluate the research outcomes, and emphasises effective communication of the research outcomes to scholastic communities and practitioners. Recommended research artifacts to represent a reality or a system, a nominal process model to establish methodological rigour in DSR, and evaluation strategy recommended in DSR projects are discussed below.

2.2.1 Design science artifacts

Design science research aims to improve the knowledge base by introducing and analysing new and innovative artifacts [184]. Hevner and Chatterjee [94] classify these artifacts into four broad categories as follows:

- *Constructs*: The concepts of a domain that describe the problem and help conceptualise the solution. Vocabulary and ontology are example of constructs.
- *Model*: A set of propositions or statements that describe the relationships among constructs. Abstraction, representation, Entity-Relationships, meta-model are the example of models.
- *Method*: A set of steps that help produce research outcome. Concept and model construction processes, algorithms, guidelines, and best practices are examples of method.
- *Instantiation*: Realisation of the constructs, models and methods as Information Technology (IT) artifacts. The IT system and prototype are examples of instantiation.

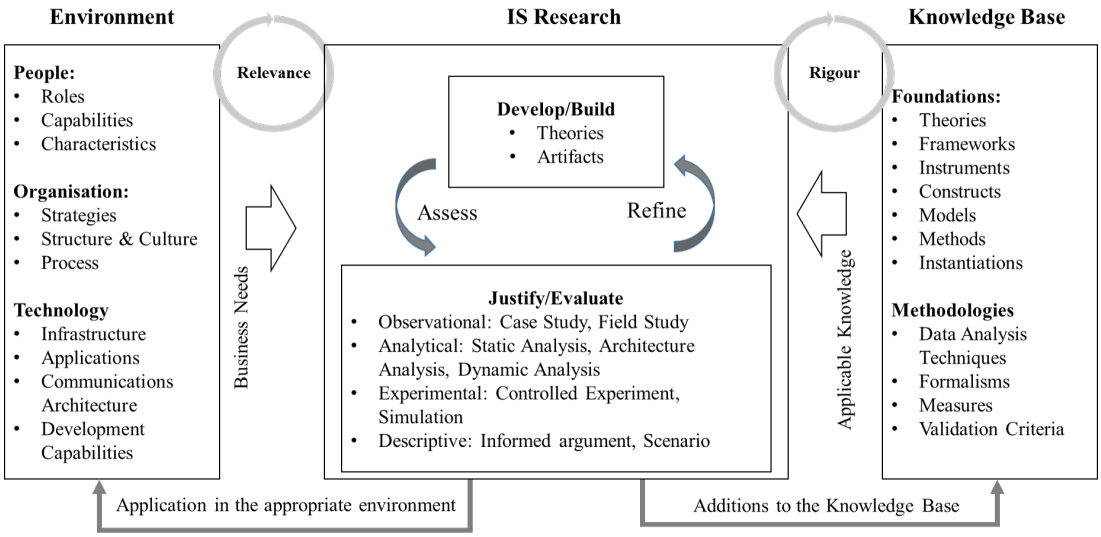


Figure 2.1 Design Science Research Framework for Information System (Source [95])

2.2.2 Design science research cycles

Figure 2.1 depicts the nominal process of DSR proposed by Hevner *et al.* [95]. As shown in the figure, the process recommends three iterative and interacting cycles: *Relevance Cycle*, *Design Cycle* and *Rigour Cycle*. The *Relevance Cycle* establishes the connection between problem space and the design science activities. This cycle iteratively defines research requirements (*e.g.*, the *Innovation*, *Improvement* or *Adaptation* opportunities), acceptance criteria to evaluate research outcomes and evaluation strategy to validate acceptance criteria.

The *Design Cycle* generates design alternatives and evaluates the alternatives against requirements defined in relevance cycle until a satisfactory design is achieved [184]. In particular, this cycle iterates between two activities - building of artifacts and its validation as depicted in Figure 2.1 with an aim that the artifacts are rigorously evaluated for the properties that are defined in relevance cycle.

The key considerations of rigor cycle are twofold - (a) ensure designs produced are not *routine design*, *i.e.*, they are either *Improvement*, *Adaptation* or *Invention*, and (b) establish produced knowledge as new scientific knowledge. As shown in Figure 2.1, the new knowledge can be broadly classified into two types - foundational contribution and methodological contribution. The foundational contributions are the *meta-artifacts* [102], *i.e.*, experiences, expertise and knowledge about the invented artifacts such as theories, frameworks and models. The methodological contribution is the knowledge that describes how an invention can be validated and effectively used by the practitioners.

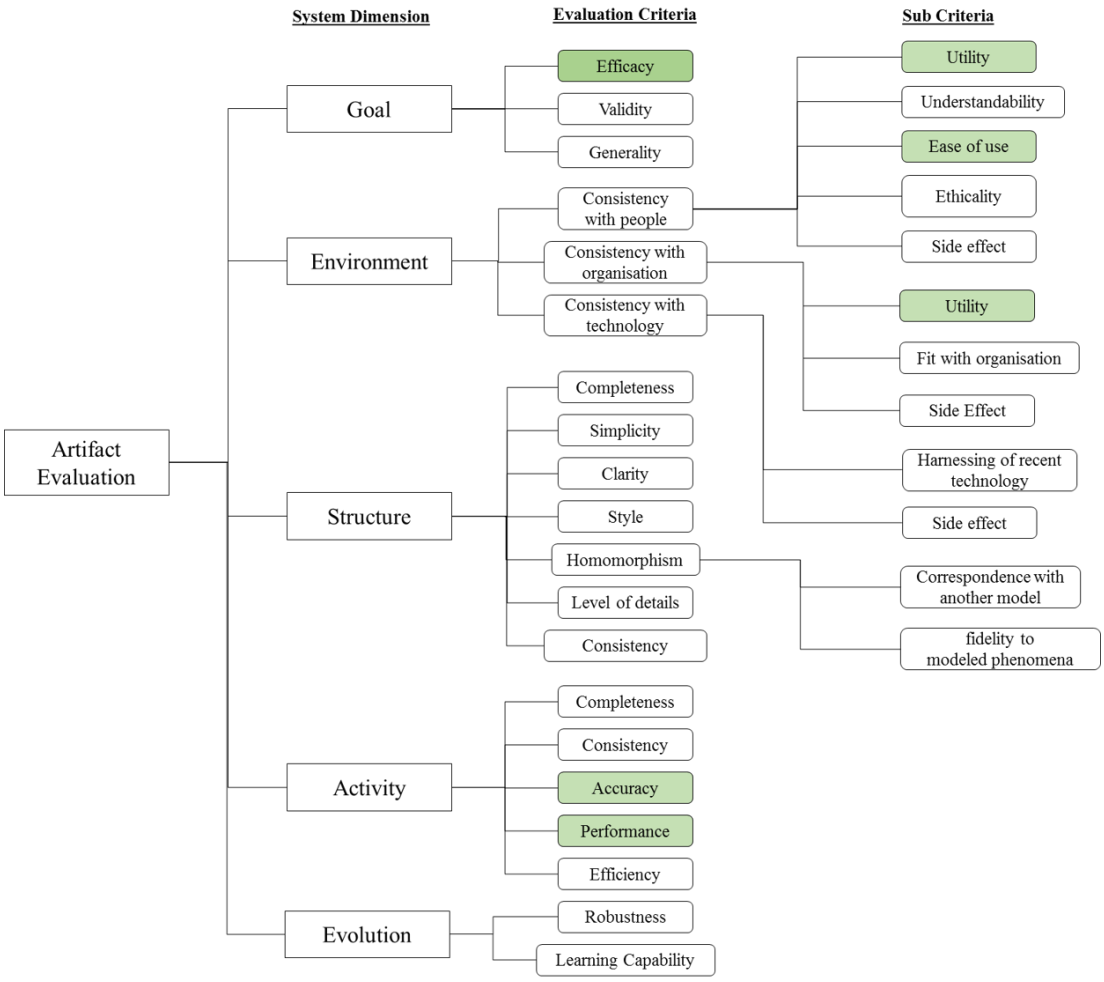


Figure 2.2 Hierarchy of criteria for IS artifact evaluation (Source Figure 1 of [160])

2.2.3 Research evaluation

A collection of research contributions from Prat *et al.* [160], Pries-Heje *et al.* [161] and Venable *et al.* [203] establish a comprehensive strategy to evaluate research artifacts produced in DSR. The strategy includes four important dimensions - (1) *why* to evaluate, (2) *what* to evaluate, (3) *when* to evaluate and (4) *how* to evaluate.

The *why* aspect or the objective of validation is to justify how well an artifact achieves its expected utility. Prat *et al.* conducted a rigorous literature review [160] and developed an exhaustive list of properties that are considered as validation criteria in a wide range of DSR literature. The hierarchy of criteria presented in [160] is depicted using a tree structure in Figure 2.2. As shown in the figure, Prat *et al.*, proposed twenty evaluation criteria along five system dimensions: *Goal*, *Environment*, *Structure*, *Activity* and *Evolution* (where presented system dimensions conform to the components of design theory, *i.e.*, purpose, scope, form,

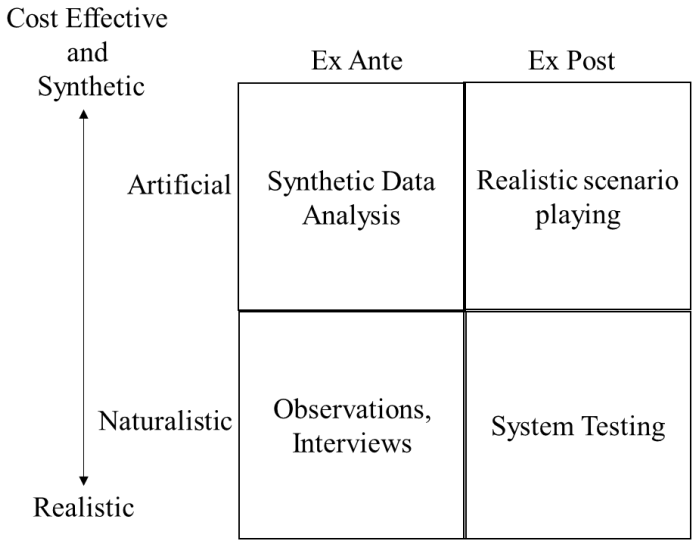


Figure 2.3 Strategic DSR Evaluation Framework (Source Figure 1 of [161]

Table 2.3 Evaluation methods

Strategy	**Method**
Observational	Case Study
Analytical	Static Analysis, Architecture Analysis, Optimisation
Experimental	Controlled Experiment, Simulation.
Testing	Functional Testing, Structural Testing
Descriptive	Informed argument, Scenario

function, artifact mutability [87]). Among these twenty properties, *efficacy* that indicates the degree to which the artifact achieves its goal is the most prominent property of interest in DSR. Venable *et al.* [203] also recommend *efficacy* as the property of interest for DSR projects. The other properties that are prominently used in DSR projects are *utility*, *ease of use*, *accuracy* and *performance* (highlighted in Figure 2.2).

Pries-Heje *et al.* [161] describe the *what* and *when* aspects of the artifact evaluation by introducing two dimensions of evaluation - *naturalistic* versus *artificial* and *Ex Ante* versus *Ex Post* as shown in Figure 2.3. Naturalistic evaluation evaluates the real artifact whereas the artificial evaluation considers artificial artifacts, *i.e.*, constructs, models, methods and instantiations. The artificial artifact is either assessed using an abstract form of the constructs/models or through instantiations. This distinction corresponds to the *Ex Ante* evaluation (uninstantiated artifact) and *Ex Post* evaluation (instantiated artifact). The dimension of *naturalistic* versus *artificial* describes *what* to evaluate and the dimension of *Ex Ante* versus *Ex Post* describes the *when* to evaluate an artifact in DSR.

DSR literature, such as [184], further proposes a range of methods and guidelines for evaluating artifacts. The high level strategy and corresponding evaluation methods recommended are listed in Table 2.3. This methodological guidelines establish the *How* aspect of the artifact evaluation.

2.3 Synthesis and realisation of DSR methodology

This research attempts to conceptualise and produce an approach and corresponding technological aid as research contributions to improve the *efficiency* [160, 203] of organisational decision-making. It posits that the problem domain, *i.e.*, organisational decision-making, is a multi-disciplinary field that includes several applied sciences, such as information systems, management, decision science and socio-technical systems. The context of the problem, *i.e.*, the organisation, is inherently complex and dynamic. Moreover, it exhibits significant uncertainty and emergent behaviour. Therefore the success of the research, *i.e.*, achieving efficiency in organisational decision-making, largely depends on multiple factors such as effective technical aids and cognitive capabilities of decision-makers. Such rationales and associated complexities lend this research to consider two broad research activities: *problem understanding* and *problem solving* as suggested by Hevner *et al.* [95] and March *et al.* [131].

Hevner *et al.* and March *et al.* recommend two complimentary paradigms: *behavioural science research* and *design science research* for problem understanding and problem solving respectively. This research adopts a design science research methodology for problem solving methodology. However, it relies on systematic literature reviews in the form of Systematic Mapping Study (SMS) [155] and Systematic Literature Review (SLR) [111] methodologies as opposed to behavioural science research for problem understanding. The key reason for such adaptation is to rely on the existing ontological and epistemological views (*i.e.*, knowledge exist in the form on publications, industrial reports and technologies) as the basis for problem understanding as opposed to a derived kwnoledge from the reality through behavioural science research.

Consistent with DSR methodology [95], the problem solving activity draws upon the *science of the artificial* [184] as a philosophical foundation. Precisely the *physical system* is visualised in terms of *models* and comprehended using *simulation* technique. As research artifacts, this research produces a conceptual schema of decision-making, a domain-specific meta-*model* to represent the physical system, an integrated *method* for model construction,

model validation and decision-making, and a technology aid as an *instantiation* of research contributions. In addition, property *efficacy* [203] is considered as a primary validation criteria to validate research artifacts; *Artificial* experiments and *Ex-Post* [161] evaluation is adopted as validation strategy; and publications, tutorials and proof-of-concept implementations are considered as the communication channels for this research. A method that combines *problem understanding* and *problem solving* activities in a seamless manner is described below.

2.3.1 Research method and activities

Peffers *et al.* [154] propose six activities to realise the nominal process presented by Hevner *et al.* in [95]. As summarised in section 2.2.2, the activities proposed by Peffers *et al.* are: *Problem identification and motivation*, *Define the objectives for a solution*, *Design and development*, *Demonstration*, *Evaluation* and *Communication*. *Problem identification and motivation* activity defines specific research problem and justifies the utility of a solution. *Define the objectives for a solution* activity defines the objectives of a solution by defining the properties to be assessed. The properties can be quantitative, *e.g.*, quantitative terms in which a desirable solution would be better than existing ones, or qualitative, *e.g.*, a qualitative description of how a new artifact is expected to address problems. *Design and development* activity produces artifact, *i.e.*, constructs, models, methods and/or instantiations. *Demonstration* activity demonstrates the utility of the artifact through experimentation, simulation, case study, proof of concept, or other appropriate method as described in Table 2.3. *Evaluation* activity validates the properties of interests. Finally, the *Communication* activity communicates utility of produced artifacts to relevant scholastic and academic communities.

This research extends the realisation method proposed by Peffers *et al.* as depicted in Figure 2.4. As shown in the figure, the extended method contains eight research activities with three iterative loops. The iterative execution of three research activities: *Problem identification and motivation*, *Exploring state-of-the-art and state-of-the-practice of organisational decision-making* and *Define the objectives for a solution* form the relevance cycle of this research. The design cycle comprises next four research activities: *conceptualisation of proposed approach*, *Instantiation*, and *Evaluate research outcome* and *Demonstration and Communication*. Finally, research activity *Establishing rigour* defines the rigor cycle of nominal process presented by Hevner *at el.* in [95].

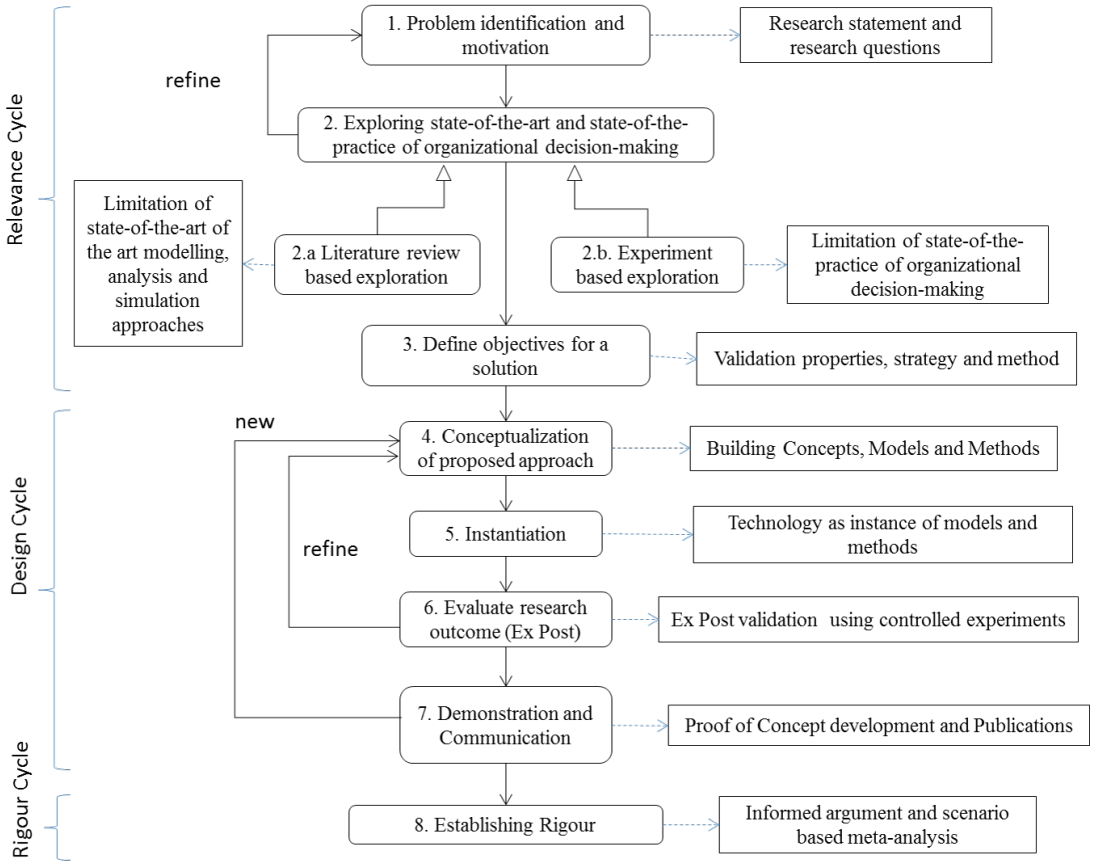


Figure 2.4 Overview of research methodology

From the perspective of realisation of DSR methodology presented by Peffers *et al.*, the research activity *Problem identification and motivation* and *Exploring state-of-the-art and state-of-the-practice of organisational decision-making* of Figure 2.4 collectively represent activity *Define the objectives for a solution* of DSR methodology. Activity *Define the objectives for a solution* and *Evaluate research outcome* of Figure 2.4 respectively correspond to the activity *Define the objectives for a solution* and *Evaluation* of DSR methodology. The research activities *Conceptualization of proposed solution* and *Instantiation* of Figure 2.4 collectively represent activity *Design and development* of DSR methodology. Activity *Demonstration and Communication* of Figure 2.4 combines two DSR activities: *Demonstration* and *Communication*. The description of eight research activities and their expected outcomes are described below:

1. *Problem identification and motivation*: A systematic literature review of management literature and industrial reports for *problem understanding*. This step produces the research statement and research questions, which are described in Chapter 1.

2. *Exploring state-of-the-art and state-of-the-practice of organisational decision-making*: Exploration of solution space, *i.e.*, techniques and technologies used in practice and the existing techniques and technologies that have potential to address the problem. This activity uses three specialised techniques: SMS [155], SLR [111] and experimentation (with modelling and analysis technologies) as a research approach to explore the capabilities and limitations of state-of-the-art modelling and analysis techniques. This step produces reports on suitability analysis of Enterprise Modelling techniques and actor technologies as presented in Chapter 4.

3. *Define the objectives for a solution*: Define evaluation property and evaluation strategy to validate property. This step identifies *efficacy* as a validation property and adopts *Artificial* and *Ex-Post* [161] as an evaluation strategy. It is *Artificial* as the synthetic case studies illustrating scenarios from industry and academia are used as validation and *Ex-Post* as the evaluation is performed on produced technology, *i.e.*, after *Instantiation*.

4. *Conceptualization of proposed solution*: Formation of ontology, model and method. This step (a) formalises concepts (ontology) of decision-making and organisation (as context) using meta-modelling techniques, (b) defines a domain-specific language to represent an organisation, and (c) proposes an integrated *method* for model construction, model validation and decision-making.

5. *Instantiation*: Develop a technology aid for domain specific language and integrated method.

6. *Evaluate research outcome*: An *Ex-Post* validation is planned where proposed approach and technology aid are used for organisational decision-making. This step considers four synthetic but close to real life scenarios from software service provisioning organisation, business process outsourcing industry, a financial disruption in India and a management side of an University as discussed in Introduction chapter.

7. *Demonstration and communication*: Demonstration using industrial exemplars and communication through scholastic publications and tutorials.

8. *Establishing rigour*: Establish connection between research outcomes and knowledge-base using meta-analyses on *Ex-Post* evaluations of *Evaluate research outcome* activities.

2.4 Summary

The philosophical grounding, research design and research methodology followed in this research are discussed in this chapter. From a broad methodological viewpoint, it justifies the relevance of two fundamental research activities: *problem understanding* and *problem solving* as proposed by March *et al.* [131]. For problem understanding, the existing epistemological foundations are explored using a combination of SMS and SLR methodologies. The design science research is adopted for problem solving. Consistent with the DSR methodology, four artifacts are proposed as research outcomes. The artifacts are – (i) the key concepts of decision-making (as *constructs* artifact), (ii) suitable schema (i.e., a meta-model) to represent complex dynamic organisation (as a *model* artifact), (iii) a simulation based methodology to explore decision-space (as an *method* artifact), and (iv) a technology aid that instantiates proposed meta-model and simulation method (as an *instantiation* artifact). The *efficacy* of decision-making is considered as desired validation criteria and an artificial *Ex-Post* validation is adopted as validation strategy in this research.

Chapter 3

Organisational Decision Making

Diverse organisations such as corporate firms, banks and government agencies are not like machines that are governed by the laws of physics or a set of mathematical equations. Herbert Simon describes organisations as *exceedingly complex* systems that are – '*made up of a large number of parts that interact in a non-simple way. In such systems, the whole is more than the sum of their parts, not in an ultimate, metaphysical sense, but in the important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole.*' [183].

The management literature on organisational decision-making proposes a range of management frameworks, approaches, methods and guidelines to understand the inherent complexities of the involved organisation and effectively deal with the dynamism and uncertainties.

This chapter reviews the characteristics of complex organisations, explores the relevant concepts and methods of organisational decision-making and discerns the key concepts in a structured form. The chapter starts with a brief overview of system theory and management view of the organisational theories in Section 3.1. It explores General System Theory (GST) [206], cybernetics [14], System Dynamics (SD) [78], Complex Adaptive Systems (CAS) [99], complexity theory [10, 7], and the Information System (IS) view of Enterprise Architecture (EA) research, such as Zachman Framework [220]. Section 3.2 discusses the prominent methods to address organisational decision-making as suggested by decision-making theorists such as James March, Herbert Simon [180, 182], Henry Mintzberg and Richard Daft [69]. Section 3.3 concludes the chapter with a precise review synthesis using a conceptual model and a set of requirements. The conceptual model describes the relevant concepts and their relationships to

describe the domain of organisational decision-making and requirements indicate the necessary capabilities to effectively address organisational decision-making.

The first two sections of this chapter explore research question *RQ1* that interrogates the necessary information for an effective organisational decision-making. Section 3.3 presents the *Contribution 1*, *i.e.*, a *Constructs* artifact (from DSR perspective) to describe organisational decision-making, as described in the Introduction chapter. With respect to the research methodology chapter presented earlier, this chapter focuses on three research activities: *Problem identification and motivation*, *Exploring the state-of-the-practice of organisational decision-making* and *Define the objectives for a solution* (*see* Figure 2.4).

3.1 Characteristics of complex organisation

The classic model of organisation as a *closed* and deterministic system has long been discredited in the management discipline [183, 69]. The organisational science that considers the organisation as a monolithic probabilistic entity, which can be specified and predicted using the established mathematical and statistical techniques, is also turning out to be less relevant in the context of a dynamic business environment [185]. Instead, the organisational theories that focus on *complex* and *open* systems are gaining importance for decision-making in modern organisation [124, 70]. This section discusses the concepts and theories of organisational science that closely characterises the modern organisations.

3.1.1 Organisation as open, complex and socio-technical system

Organisational theory [183, 10, 7] visualises the modern organisation as *open* system. An organisation is a *system* as it consists of interconnected components that work together [183]. It is *open* as it exchanges messages and resources with its environment. An organisation contains multiple continuous feedback loops with its environment for survival and success [10]. This system theoretic view of the organisation is primarily derived from the study of General System Theory (GST) [206] and cybernetics [14].

The foundation of *complexity theory* advances the canonical form of an organisation model. In particular, theorists consider the organisation as a *complex* entity because an organisation typically composes a large number of interdependent subsystems or elements [10, 69, 7] in a *nonlinear* way [49]. John Casti associates the *non linearity* as a primary cause for the organisational complexity. Thietart and Forgues [191] reflect on the *butterfly effect* of the *chaos*

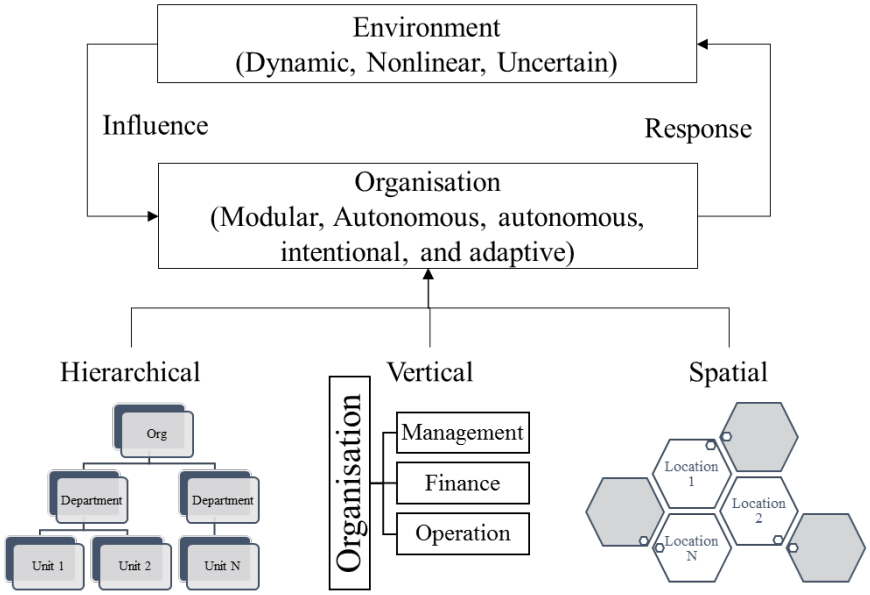


Figure 3.1 System theory view of complex organisation

theory to emphasise the extreme effect of non-linearity in the organisational context. They further demonstrate how the accumulation of several small changes in the environment or in a subsystem results into an explosive situation [191]. Daft and Lewin relate the *loose coupling* and *autonomy* of the individual elements as a prime factors for the organisational complexity [69]. Richard Daft emphasises the complex organisational structure along three dimensions, *i.e.*, *vertical*, *horizontal* and *spatial* [68], as a key cause for organisational complexity. To that view, the organisational hierarchy forms the vertical complexity, the functional units and departments form the horizontal complexity, and the distributed geographical locations of the organisation characterises the spatial complexity of an organisation as depicted in Figure 3.1.

The study of the CAS [99] further enriches the understanding of the organisation model. In principle, *complex adaptive systems* are not deterministic automatons, rather their behaviours emerge from the interactions of the connected sub-systems, individuals or *agents*. A complex adaptive system evolves over time by changing linkages between the agents, shifting the pattern of interconnections, and changing the individual agent behaviours. Moreover, the individual sub-systems, elements or agents *self-organise* [74] their structure and behaviour. These characteristics emphasise the *emergent behaviour*, *adaptability* and *autonomy* of the organisations and their constituent units.

Finally, modern organisational theory, mainly the research presented by Richard Daft [68], investigates the characteristics of the system elements that include the sub-systems, elements

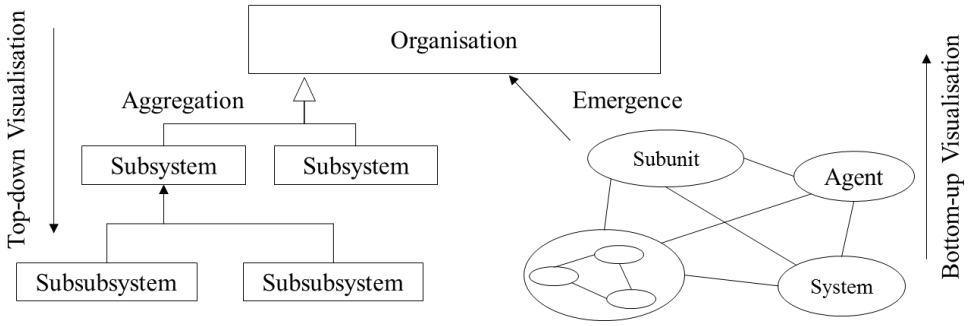


Figure 3.2 Top-down and bottom-up visualisation

and/or agents. The theory advocates a socio-technical viewpoint to describe the system elements. It considers the organisations are made up of multiple mechanistic and organic entities, wherein the mechanistic entities are not necessarily the machines but they are characterised by machine-like standard laws and rules. The organic entities, in contrast, are free-flowing and adaptive. They do not strictly conform to mathematical equations or law of physics, rather they are flexible to adopt behavioural rules and regulations at any given time. This socio-technical viewpoint acknowledges the *adaptability* and *autonomy* as described in CAS theory. In addition, the socio-technical viewpoint imparts the *uncertainty* and *intentionality* as inherent characteristics of the organisation.

3.1.2 Philosophical viewpoints for system understanding

From a philosophical perspective, reductionism and holism are two viewpoints that help investigate a complex system or an organisation [34]. The reductionists claim a complex system, such as an organisation, can be comprehended by understanding all its constituent parts, whereas the holistic tradition tries to comprehend an organisation as a whole. Organisational theory argues that the pure form of reductionism is less effective as a viewpoint to understand a complex system [10] as the impact of the interconnections, feedback loops, and emergent behaviour are grossly neglected if the parts or subsystems are studied in an isolation [34]. Therefore, the theorists suggest a holistic view to understand the complex system with feedback, uncertainty and nonlinearity.

In addition, top-down and bottom-up approaches can be used [192] as illustrated in Figure 3.2. The *top-down* approach visualises a system from a higher scale and focuses on an aggregated macro-behaviour. Primarily, it adopts a belief that the constituent micro-behaviours of a system can be reasonably approximated as a macro behaviour or set of macro-behaviours.

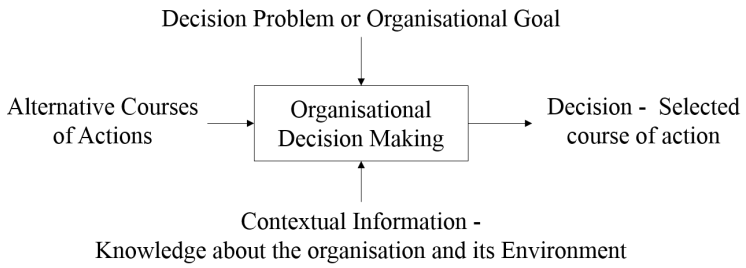


Figure 3.3 High level schema of organisational decision-making

The System Dynamics (SD) model [78] and Enterprise Modelling (EM), such as Zachman Framework [220] and ArchiMate [100] are popular computer models that support the top-down approach. The bottom-up approach, in contrast, focuses on the micro-behaviours and attempts to understand how multiple micro-level interactions drive the macro-behaviour of a complex system, *i.e.* emergent behaviour. Cellular automata [148] helps to understand the behaviour that emerges from the spatial composition and neighboring interactions, whereas the *agent* based models [178, 47, 129] and *actor* model [2, 96] help to investigate a wide range of generic bottom-up behaviours. Chapter 4 discusses a range of computer models and their suitability in modelling and analysis of complex organisation in details.

3.2 Characteristics of organisational decision making

Organisational decision making is a process of selecting a belief or a course of action among several alternative possibilities for achieving organisational goals [130]. Principally, a decision, *i.e.*, a course of action, is a consequential action that influences an organisation to grow, prosper, or fail at varying range of significance over the time horizon [70]. The management perspective defines, characterises and classifies these organisational decision-making processes and proposes appropriate styles, methods and approaches to address a range of decision-making problems. This section highlights decision-making concepts, classification schemes, approaches and decision-making models.

3.2.1 Core concepts of organisational decision-making

The decision-making literature [180, 62, 108, 177, 8] considers four broad concepts to describe organisational decision-making as illustrated in Figure 3.3. The *decision problem* or *organisational goals* is the objective, *contextual information* is the relevant knowledge about the

organisation and its environment, *courses of action* are the decision alternatives that decision makers consider and the *decision* is a selected alternative, which is an outcome of a decision making. James March formalises organisational decision-making [130] using four primitives as follows:

- *Knowledge of alternatives*: A set of belief or courses of action that has the potential to address the decision problem. The alternative may include an *inaction*.
- *Knowledge of consequences*: Possible consequences of the alternative courses of action.
- *Consequent preference ordering*: The variables by which the consequences of the courses of action can be compared.
- *Decision rule*: Rules by which decision-makers rank and select a course of action from a list of alternatives.

Theoretically, the *Knowledge of alternatives* represents the alternative courses of action, the *Knowledge of consequences* simplifies the analysis or interpretation of the *contextual information*, *Consequent preference ordering* represents the measurable variables that indicate whether a *decision problem* is solved and/or the *organisational goals* are achieved, and the *Decision rule* represents the rules for ranking and selecting alternative from available options by knowing the *Knowledge of consequences*.

In this framework, the simplest form of decision-making is a situation where all the alternatives are precisely known, the consequences are known or can be precisely computed/predicted from the available contextual information, and the decision rule is the selection of an alternative with the highest expected utility value (from economical perspective). However, the expectation of complete *contextual information*, *i.e.*, the information necessary for an organisational decision-making, is a difficult proposition. There are many specifics that are unaware, unknown or uncertain in a typical organisational context [62]. For examples, interactions within organisational units can be probabilistic, environment where an organisation operates can be uncertain, and goal of a decision problem can be unclear at the beginning of a decision-making.

These unknown factors and uncertainties are well studied in management literature and they are precisely described using four terms [70], namely: *certainty*, *risk*, *uncertainty*, and *ambiguity*. The *certainty* is a situation where the necessary information for a decision-making is known; *risk* is a situation where there is a degree of uncertainty in predicting the possible

consequences for known alternatives; *uncertainty* describes a situation where organisational goals are known but the contextual information of a decision-making is largely unknown. The *ambiguity* of a decision-making is a difficult decision situation where the problem to be solved or the goals are unclear, the alternatives are difficult to know, and the information about the consequences are unavailable. Richard Daft further defines an extreme ambiguous situation as a *wicked decision problem* [70] where the goals are conflicting, the decision alternatives are difficult to imagine, environment is fuzzy, and contextual information about the organisation is incomplete and fragmented.

Based on the level of uncertainty, the organisational decision-making is classified into different categories along multiple dimensions. Different classification schema and classifications are described in the next subsection.

3.2.2 Classification of organisational decision-making

Conceptually, organisational decision-making is classified into two broad categories - *programmed* decision-making and *nonprogrammed* decision-making [67]. The programmed decision-making addresses the decision problems, which are certain and associated with low risk consequences. The alternatives are known, consequences of the alternatives are not significantly high, *decision rules* for all possible alternatives are fully known (or can be derived from historical occurrences), and the context is relatively static and mechanistic.

The *nonprogrammed* decision-making, in contrast, addresses the scenarios that are new or novel, and they exhibit significant uncertainty and/or ambiguity. The *nonprogrammed* decision making deals with the situation, where the alternatives, their consequences, and the decision rules cannot be inferred from the historical occurrences. The strategic decision-making of modern organisations are mostly *nonprogrammed* decision-making in nature. Decision practitioners adopt different methodological viewpoints to approach these decision-making problems.

From a theoretical perspective, organisational decision-making follows a usual dichotomy that distinguishes between the *normative* and *descriptive* decision-making styles, and considers a third perspective - the *prescriptive* decision making style [35]. A normative decision making style defines how a decision ought be made and provides guidelines for ideal decision-making situation. The descriptive decision-making describes how decision makers make the decision rather than what ought to be done in an ideal situation. On the hand, the prescriptive decision-making combines both the styles by exploiting the normative theories, and adopting the useful

observations of the descriptive decision-making. These viewpoints get reflected in management classification that ranges from an *economically rational* model to a *bounded rational* model to an *anarchy* model. From a management perspective, the organisational decision-making falls into one of the four primitive decision-making models: *Management science* [8], *Carnegie Model* [66], *Incremental Process model* [141] and *Garbage Can model* [60]. The primitive models and their characteristics are described below:

Management Science model

The management Science model (also known as Operational Research (OR)) [8] is a rational and normative approach that works under following assumptions

1. Decision problem or goals are precisely known.
2. Alternatives are known and bounded.
3. Potential consequences of the alternatives can be computed from available information.
4. Decision rules or criteria for ranking alternatives thrive to maximise the economic return.

This approach is an effective organisational decision-making model when a problem is analyzable, the context can be represented using mathematical models, and the possible consequences can be computed using mathematical and statistical rigour. This model is most useful for the programmed decision-making.

Carnegie model

The Carnegie Model [66] is based on the work of Richard Cyert, James March, and Herbert Simon undertaken at Carnegie University as an extension to the *administrative behavioural model* [180]. It is a descriptive form of decision-making, where decision-makers actually make decisions for the complex situations as opposed to dictating how the decision should be made in a theoretical ideal. The basic assumptions of the Carnegie Model are:

1. Decision goals are uncertain and ambiguous.
2. Alternatives are partially known.
3. Probable consequences are conflicting.

4. There is no fixed decision rules to select alternatives.

The Carnegie Model approaches decision-making based on two philosophical beliefs - (i) *bounded rationality*, and (ii) *satisficing* (a term coined by Herbert Simon to describe satisfy and suffice). The bounded rationality belief discusses how rational the decision makers can be in a situation where the organisation is incredibly complex, there are infinitely many alternatives, and the decision makers have limited time and processing capability. The concept *satisfice* suggests choosing the first alternative that achieves the organisational goals as opposed to exploring all options and select an economical optimum solution.

The model proposes discussion, negotiation, coalition and bargaining as possible management aids to resolve ambiguous goals, conflicting beliefs about the possible consequences, and inadequate decision rules [92].

Incremental Decision Process model

Henry Mintzberg et al. [141] approach a decision-making using a structured sequence of activities that make a series of small choices to produce a significant decision. The basic assumptions of the incremental decision process model are:

1. Decision problem or goals are well defined.
2. Alternatives are not known but the high-level line of attack in terms of fine-grained courses of action are known.
3. Consequences are not known but the consequence of fine-grained courses of action can be predicted or computed from available information.

This approach focuses on an iterative and incremental approach to solve a significant decision problem (as opposed to the political and social considerations such as collaboration and bargaining as described in the Carnegie Model). The process starts with an assumption or a line of attack, and moves through various smaller decision points that consider the fine-grained alternatives. The process may face the barriers where it cycles back to a previous decision point, and try a new alternatives. These barriers are known as *decision interrupts* and the cycle exploring a series of alternatives are known as *decision loop* [123]. The decision makers learn the possible consequences through multiple decision interrupts and decision loops.

Table 3.1 Decision Making Approaches

Model	Characteristics	Decision Style	Analysis Style	Analysis Technique
Management Science [8]	Problem and solution spaces are well defined	Normative	Formalize problem statements as mathematical formulae to be solved using appropriate algorithm.	Bayesian statistics, PERT charts, linear programming
Carnegie Model [66]	Uncertainty about problem space	Descriptive	Funnel in all relevant information and rely on decision makers to make choices.	Discussion, Interview, workshop, bargaining , negotiation and conflict resolution.
Incremental Process model [141]	Certain about problem space but uncertain about solution space.	Prescriptive	Solve big problem using many incremental judgmental decisions.	Judgmental call or prediction based on past experience, data, or intuition.
Garbage Can model [60]	Problem and solution space both are uncertain.	Prescriptive	Organised anarchy.	Intuition based

Garbage Can model

The garbage Can model is introduced by Michael Cohen, James March, and Johan Olsen [60] to capture the most uncertain and ambiguous situations of the organisation. The basic assumptions are:

1. The decision problem and goals are ambiguous.
2. Alternatives and decision rules are ill defined.
3. The contextual information is unclear and difficult to understand.

The Garbage Can model considers several atypical characteristics of the organisation decision-making. For example, the decision can be made even when the problem does not exist, the choices can be made without solving an actual problems, or some other problem may get resolved while solving a problem. The fundamental belief behind the Garbage Can model is only a few problems are actually solved in any decision-making initiative but the organisation should keep working towards problem reduction. The management viewpoint suggests an anarchy to aim for such unplanned but useful improvements.

Summary

The high-level characteristics, recommended analysis style and the state-of-the-practice of analysis techniques for four decision-making models that are discussed in this section are

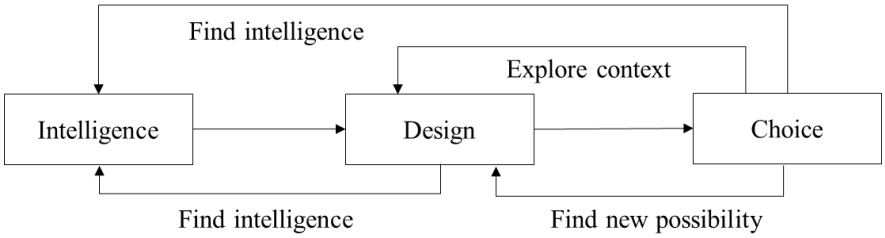


Figure 3.4 Organisational decision-making process proposed by Herbert Simon [181]

summarised in Table 3.1. As it can be noticed, the organisational decision-making models discussed in this section are limited to the management literature that excludes social, political and psychological aspects. For example. decision makers can be politically and/or socially biased in a decision-making process [157]. The literature that focuses on how to deal with such situations are not considered in this thesis.

3.2.3 Organisational decision-making processes

The management literature advocates the methodological rigour for organisational decision-making. Herbert Simon first proposed an iterative process in [181]. As shown in Figure 3.4, the proposed process considers three phases: *Intelligence*, *Design* and *Choice*. The intelligence phase defines the problem statement. The design phase investigates the context of the decision-making, *i.e.*, organisation and its environment, and develops possible alternatives. The choice phase selects the most appropriate alternative from the available alternatives. Each phase of the decision-making process may itself form a decision making process. For example, a design phase may initiate an intelligence phase, and a choice phase may trigger an intelligence phase followed by a design phase for various reasons as depicted in Figure 3.4.

Herbert Simon subsequently extended his essential process model with two additional phases: *Implementation* and *Review*. The implementation phase implements the selected decision, and the review phase evaluates the efficacy of the selected decision. Richard Daft further extends Simon's decision-making process by taking consideration of different categories of decision-making (that includes the *programmed* and *nonprogrammed* decision-making) and range of organisational decision-making models (that includes the Management Science, Carnegie Model, Incremental Process model and Garbage Can model). A generic organisational decision-making process is proposed by Richard Daft in [70] is pictorially shown in Figure 3.5. The specific activities of the process steps are described below:

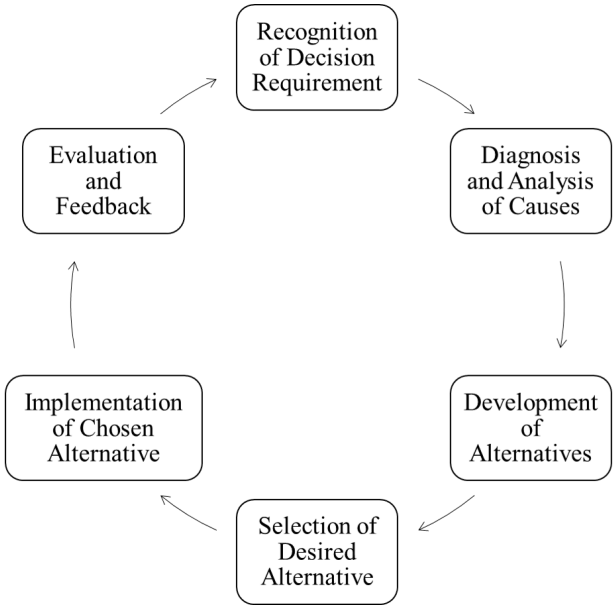


Figure 3.5 Organisational decision-making process proposed by Richard Daft [70]

1. **Recognition of Decision Requirement**: A decision-making process typically starts with a decision requirements in the form of either a problem or an opportunity. A problem occurs when the performance of an organisation is less than its expectation or the organisation is unable to achieve its desired goals. Whereas an opportunity is a potential improvement initiative.

2. **Diagnosis and Analysis of Causes**: This step analyses the *context* and the underlying causal relationships associated with the decision situation. It makes an attempt to understand the inherent characteristics, current state, operating environment, and inherent dynamism of the organisation.

3. **Development of Alternatives**: This step generates possible alternative solutions or courses of action. For the programmed decision-making and the Management Science model, all feasible alternatives are expected to be known from the historical instances or through intuition. The *nonprogrammed* decision, however, expects a set of new courses of action. For Carnegie Model, the decision makers start with limited alternatives to *satisfice* the decision-problem. The Incremental Process model develops small choices that can be sequenced to form the potential courses of action. The Garbage Can model expects an innovative way to conceptualise new alternatives.

Table 3.2 Decision step of organisational decision-making

Decision Type	**Recognition of Decision Requirement**	**Diagnosis and Analysis of Causes**	**Development of Alternatives**	**Selection of Desired Alternative**
Pro-grammed	Complete certainty about the problem statement	Uses mathematical, statistical and/or OR approaches	Bounded set of alternatives	Optimum solution is derived
Non-programmed	Problem statement can be uncertain and ambiguous	Uses intuition, mathematical approach, simulation	Alternatives are not bounded	Solution may not be optimum
Manage-ment Science	Complete certainty about the problem statement	Uses mathematical, statistical and/or OR approaches	Bounded set of alternatives	Uses optmisation algorithm
Carnegie Model	Problem statement is not certain	Uses intuition based approach. Exploit social and political viewpoint	Alternatives are not bounded	Solution *satisfice* the decision-problem
Incremental Process model	Defines precise problem statement	Uses incremental method to understand context	solution is selected from sequence of micro-steps and validations	Solution may not be optimum
Garbage Can Model	Problem statement can be uncertainty and ambiguous	Uses intuition, mathematical approach, simulation	Alternative is not bounded at the begining	Solution may not be optimum

4. **Selection of Desired Alternative**: This step makes a decision choice by selecting the most promising course of action. Decision maker tries to select a choice with least *risk* and *uncertainty*. The Management Science model of decision-making uses mathematical rigour, Carnegie Model relies on the intuitions and experiences gained from discussions and interviews, and the Incremental Process model effectively combines the mathematical rigour and intuition.

5. **Implementation of Chosen Alternative**: The success of a chosen alternative depends on the selection of an effective course of action, and its successful implementation. The step adopts communication, motivation, and leadership skills to resolve implementation related uncertainties and ambiguities.

6. **Evaluation and Feedback**: This step evaluates the effectiveness of the chosen decision in achieving the organisational goals, and generates feedback describing the efficacy and/or inefficacy of a decision and its implementation. The feedback may initiate a new decision-making process as depicted in Figure 3.5

Though the above decision-making process steps are common for the *programmed* / *nonprogrammed* decision-making and applicable for all four decision-making models, the activities

within the first four steps differ significantly based on decision-making classifications. Table 3.2 illustrates the fundamental differences for *programmed* decision-making, *nonprogrammed* decision-making, Management Science model, Carnegie Model, Incremental Process model and Garbage Can model. It can be observed from the table, the activities for *programmed* decision-making and the Management Science model are same as they focus on similar class of organisational decision-making problems whereas the *nonprogrammed* decision-making conceptually represents the management classification of Carnegie Model, Incremental Process model and Garbage Can model.

3.3 Review synthesis and requirements derivation

As highlighted in the earlier sections, the management literature establishes a strong theoretical, conceptual and empirical foundations to address a wide range of organisational decision-making problems. However, the effective utilisation of the technological aids are largely limited to the programmed decision-making and the classical management science decision-making problems. This research highlights an opportunity to develop suitable technological aids to address a wider range of organisational decision-making problems. The research hypothesises a suitable technological support to capture and analyse necessary information to explore the *knowledge of alternatives*, enrich the *knowledge of consequences* with evidences, and develop effective *decision rules* for unforeseen situations in the face of increasing *complexity* are key for such wider adoption.

This section synthesises the knowledge of organisational decision-making and develops two research artifacts - (i) a new conceptual model that describes the key concepts of the organisational decision-making, and (ii) modelling and analysis needs to capture and analyse the required information. From design science research methodology perspective, the proposed conceptual model is a *Constructs* artifact [94] of this research. The proposed *Constructs* help define the problem space. The modelling and analysis needs serve as the requirements to explore the suitability of existing technologies and ascertain the required technological advancements.

3.3.1 Conceptual model

The theoretical foundations and the management viewpoints identify a set of concepts to describe the organisational decision-making that are shown using a class diagram in Figure 3.6. The models [8, 66, 141, 60] and the decision-making processes [181, 68] consistently use three

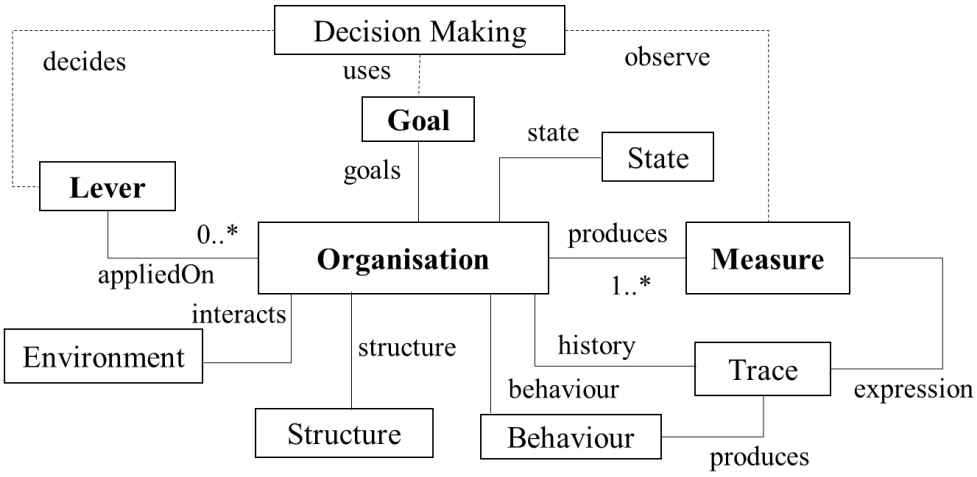


Figure 3.6 Meta model of organisational decision-making

concepts that principally describe - (i) the decision objective, (ii) the courses of action, and (iii) the variables by which the consequences of courses of action can be measured. These concepts are represented using three conceptual entities: `Goal`, `Lever`, and `Measures` in Figure 3.6. The concept `Goal` represents the organisational goals as described in [68], the *Intelligence* as described in [181], and the decision problem that triggers a decision-making process [69]. The `Lever` is a conceptual representation of a course of action or an alternative from the *knowledge of alternatives* [130]. `Measure` represents the *key performance indicator* (KPI) [68] or the variables that form the *Consequent preference ordering* [130].

Key organisational decision-making literature [177, 62, 185, 126, 180, 130] acknowledges that a decision-making cannot happen in vacuum. It requires the *contextual information* to (a) compute or predict the consequences of the possible courses of action as suggested by Richard Daft in [70], (b) develop the *knowledge of consequences* as recommended by James March in [130], or (c) carry out the analysis required for *Design* phase of the process model defined by Herbert Simon in [181]. The concepts `Organisation` and `Environment` are introduced in the proposed concept model to represent the contextual information.

Modern organisation theory, such as [108], visualises an `Organisation` as an *open system* (in consistent with GST as discussed in section 3.1.1). The notion of a system that has the structure, behaviour and state is explicated using three concepts: `State`, `Structure` and `Behaviour`. In addition, the perception of an open system is imparted using the `interacts` relationship between the `Organisation` and `Environment` as shown in Figure 3.6. Organisational theory considers a notion of an *organisational memory* [124] that records historical states, interactions,

realisation of goals, and the useful phenomena. The concept `Trace` is introduced in the proposed concept model to represent the relevant information of *organisational memory*.

The concepts `Goal`, `Measure` and `Lever` are derived from the methodological aspect of the organisation decision-making, and the concepts `Organisation`, `Environment` and their descriptive elements, such as `Structure`, `Behaviour`, `State` and `Trace`, are derived from the organisational theory that includes the general systems theory [206], cybernetics [14], and the complexity theory [10, 7]. The two sets of conceptual definitions converge at the concepts of `Lever` and `Measure`. The courses of action or `Levers` are the changes or the transformation functions over the elements of `Organisation` that includes the `Structure`, `Behaviour`, and `Goal`. `Measures` are expression over an organisational memory, *i.e.*, `Trace` and `State` of the `Organisation`.

In this formulation, the organisational decision making can be defined as a process to develop the *Knowledge of consequences* [130] by computing/predicting the `Measures` for all `Levers` (*i.e.* the *Knowledge of alternatives* [130]), rank the `Levers` based on the `Measure` values (*i.e.*, *Consequent preference ordering* [130]), and select a `Lever` based on the *Decision rules*. The *programmed* decision-making is the situation where `Levers` and `Measures` are finite and bounded, the `Organisation` and its interactions with `Environment` are deterministic, and they can be fomalised using mathematical and/or statistical models. The *nonprogrammed* decision making is a range of situations where there can be uncertainty and ambiguity in `Levers`, `Measures`, `Organisation` and/or `Environment` formation. Similarly, the Management science model of decision making can be formed by constraining the `Goal`, `Measure` and `Lever` as known and finite sets. In Carnegie Model [66], the `Goals` are uncertain, and the `Structural` and `Behavioural` aspects of the `Organisation` are not precisely defined or known. In Incremental Process model [141], the `Levers` can be visualised as a composable entity. In contrast, the fundamental assumption of Garbage Can model [60] is the `Goal`, `Lever`, `Measure` can randomly evolve in a decision-making process.

The derived concept model, which is depicted in Figure 3.6, captures the necessary *concepts* to describe organisational decision-making. This concept model can be considered as a meta-model to capture necessary aspects of organisational decision-making.

Table 3.3 Modelling and analysis requirements for effective organisational decision-making

	Requirements	Description
Specification Capabilities	Why	Information on the intention or goal
	What	Structural information
	How	Behavioural information
	Who	Information on stakeholders and responsible human actors
	When	Temporality in behaviour
	Where	Information about the location or spatial information
Organisational Characteristics	Modularity	The organisation is a set of self-controlled units
	Composability	Unit and Organisation can be assembled from units
	Reactive	Unit and Organisation must respond appropriately to its environment
	Autonomous	An unit is responsible for its own behavior and it can produce output without an external stimulus
	Intentional	Unit must have intent and it behaves accordingly to achieve its intent
	Adaptable	Unit can adapt itself based on context and situation
	Uncertainty	It is not necessary that all information about an unit will be known
	Temporality	Indefinite time-delay between an action and its response
Concepts	Goal	Ability to capture goals
	Measure	Ability to capture the Measures
	Lever	Ability to capture the courses of action or Lever
Analysis Aids	Top-down & Bottom-up	Support for top-down and bottom-up nodelling and analysis
	Emergentism	Ability to understand the emergent behaviour of interacting units
	What-if analysis	Ability to carry out what-if scenario playing

3.3.2 Tenets of organisational decision-making

An effective organisational decision-making depends on two factors: (i) the ability to capture the core decision making concepts such as the one depicted in Figure 3.6, and (ii) the ability to perform required analyses on the available information. The former requires completeness and expressibility, and the latter expects the efficacy in analysis capabilities. A list of detailed specification/modelling and analysis requirements is presented in Table 3.3.

The requirements are classified into four groups: *specification capabilities*, *organisational characteristics*, *concepts* and *analysis aids*. The *specification capabilities* group explicates organisational aspects derived from Zachman Framework [220] (which is considered as the basis for all Enterprise Modelling (EM) researches [100, 204, 93, 115, 31]). All six interrogative aspects: *why*, *what*, *how*, *when*, *where*, and *who* are considered to understand an organisation as a whole. *Organisational characteristics* group discusses eight organisational characteristics by reflecting on the theories that focus on organisational complexities as discussed in Section 3.1. *Complexity theory* visualises an organisation as set of intentional, reactive, autonomous, and adaptive units. The organisational theory discuss the possibility of the structural and the behavioural uncertainty in an organisation. Cybernetics and organisational theory highlight the

existence of nonlinearity and temporal delays in the interactions within the organisational units and between the organisation and its environment. The *concepts* group represents decision-making concepts derived from management literature.

From a methodology perspective, capturing necessary information and analysing them witness a curious dilemma between the top-down versus bottom-up [192] and the holistic versus reductionists viewpoints [34] as discussed in section 3.1.2. In an organisation, the goals mostly follow a top-down path where the top level goals are decomposed into various unit level goals along the organisational structure. However, describing the overall behaviour of an organisation in the face of increasing complexity and uncertainty is a difficult proposition. The behaviour is known only for highly localised contexts, which suggest a bottom-up modelling approach. The reductionist viewpoint helps to reduce the complexity but not able to recognise the emergentism, whereas the holism helps to understand the emergent behaviour but does not provide suitable guideline to address associated complexity. Therefore, a middle-out approach that combines the top-down and bottom-up approaches, an ability to understand the emergentism, and an ability to perform *what-if* analysis on available information are considered as the expected *analysis aids*.

3.3.3 Illustration of concepts and characteristics

The concept model presented in this section (and pictorially represented in Figure 3.6) and the organisational characteristics discussed in Table 3.3 are illustrated using a *department*, say *CS Department*, of *ABC University*, which is introduced in Section 1.6 of Chapter 1.

As discussed in section 1.6 and pictorially depicted in Figure 1.2, consider a goal of *CS Department* is to improve the research and teaching ranking with respect to the other departments within and outside university. From operational perspective, the department offers a set of *Courses*/*Modules* and involves in research activities. Structurally the department is formed with three kinds of *Academics*: *Research Academics*, *Teaching Academics* and *Research and Teaching Academics*, and it has a set of *Students* who enroll for the courses.

The organisational decision-making concepts (*i.e.*, the concepts from conceptual model introduced in Figure 3.6) of *CS Department* is illustrated in Figure 3.7. As shown in the figure, *CS Department* is a `Department` of *ABC University* and it has a `Goal` of '*Improving Research and Teaching Ranking*' from previous years. Simplistically in UK, the teaching ranking is measured using *NSS Scores* and research ranking is often determined by multiple factors such as publication counts. Therefore, the '*NSS Score*' and '*Publication Count*' of an academic year

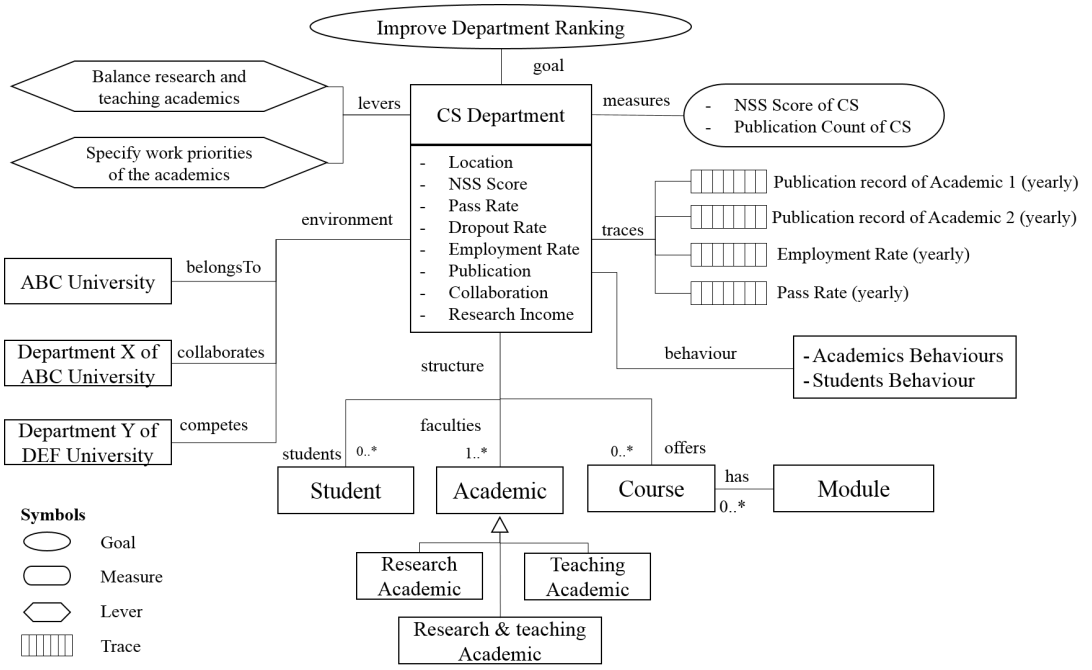


Figure 3.7 Illustration of conceptual model using university case study

can be seen as the `Measures` for specified `Goal`. Similarly, there is evidence that balancing research and teaching activities, and setting appropriate work priorities of the academics might help improve the teaching and research qualities [128]. Thus, '*Balance research and teaching academics*' and '*Specify work priorities of the academics*' can be considered as `Levers` for the *CS Department*.

From operational perspective, the *CS Department* operates within *ABC University* and competes with other `Departments` hence *ABC University* and other `Departments` (within and outside university) form the `Environment`. The department is characterised by a set state variables, such as '*Location*', '*NSS Score*', '*Pass Rate*', '*Dropout Rate*', '*Employment Rate*', '*Publications*', '*Collaborations*', and '*Research Income*' - a snapshot of these state variables at a given time defines its `State` of *CS Department*; it has a dynamic structure that includes `Academics`, `Students` and `Courses` (as the topology may change over time); and it exhibits behaviour that largely relies on the behaviours of the `Academics` and `Students` (the behaviour of *CS Department* emerges from the behaviours of `Academics` and `Students` and their interactions). The *CS Department* maintains a set of records, such as the publication records of the individual `Academics`, yearly employment rate of `CS Department`, and the yearly pass rate of `CS Department`. They form the `Traces` of the *CS Department*.

These conceptual aspects conform to the specification capabilities described in Table 3.3. The ability to capture the structure of *CS Department* defines the need for *what* aspect, and the behaviours of the active structural elements define the need for *how*, *who*, *when* and *where* aspects. The `Goal` of an organisation is suggestive of '*why*' aspect; `Measure` specification requires '*what*' information needs to be captured '*when*' and '*how*'; whereas the `Lever` specification expects the information about '*what*' changes are required/expected '*when*' and to '*where*' for achieving organisational `Goal`.

The elements described in this example exhibit the organisational characteristics discussed in Table 3.3. For example, the `Departments`, `Academics` and `Students` are self-content elements - thus they are *modular* units. They are *intentional* elements as they have their own intentions and their behaviours largely are governed by their intentions. The `Academics` can dynamically form the research `Projects`, therefore they are *composable*. The `Academics` and `Students` are *reactive* elements as they can react to the environmental events such as *Call for Paper* and *Call for Research Proposal*. They can continue their activities or interrupt current activity without any external stimuli, therefore they are *autonomous* elements. The department may change the topology over time by adding and/or eliminating `Academics` and `Students`; similarly `Academics` and `Students` may change their behaviour over time - therefore they are *adaptable* elements.

There are many events in a department that are governed by the fixed schedule (they are independent of the states of the individual elements). The admission schedule, assessment schedules, paper submission deadlines are such kinds of events. Similarly, there are some known and/or indefinite time delays between some of the organisational events. For example, time between paper submission and the notification, submission of research grant proposal and the notification, and assessment period of a module are such examples. These kinds of fixed schedules and time delays form the *temporal* behaviour of the *CS Department*.

On other hand, the `Academics` and `Students` exhibit different levels of *uncertainties*. For examples, an `Academic` shows a probabilistic behaviour to choose an activity (from the set activities as discussed in Table 1.1) at a specific time slot or interrupting the current activity to take up another activity. For a research academic, there is a varying probability amongst the individuals to write paper (from current research) or focus on more research activity for better publications in the future. Similarly, there is an inherent randomness to know an individual student will study in the free hours or not.

Table 3.4 Requirements mapping

	Requirements	Examples
Organisational Characteristics	Modularity	The organisation is a set of self-controlled units
	Composability	Department composes academics and students
	Reactive	Academics react to various events such as student queries and complaints. Similarly students react to lectures and assessments
	Autonomous	Academics and students are autonomous units as they behave their own way
	Intentional	Academics and students have their own intents
	Adaptable	Academics and students can adapt themselves based on context and situation
	Uncertainty	Academic and students may choose any activity from the activity list (as discussed in Table 1.1) at a given time
	Temporality	A student may raise a query or complaint any time after a lecture or assessment. Similarly, an academic can respond to a student query within a time range
Concepts	Goal	Improving research and teaching ranking
	Measure	NSS Score, Publication Count
	Lever	(i) Balance research and teaching academics, (ii) Specify work priorities of the academics
Analysis Aids	Top-down & Bottom-up	The goal of a department propagate to the academics in a top-down manner whereas the behaviour can be specified using academics and students (*i.e.* bottom-up manner)
	Emergentism	The behaviour of a department emerges from the behaviour of academics and students
	What-if analysis	*What* would be the situation of a department *if* academics are directed to focus more on teaching activity than research activity?

This case also demonstrates a scenario of emergent behaviour. The behaviour of *CS Department* cannot be specified as whole - the behaviour emerges from the behaviour of constituent `Academics` and `Students`, associated `Environment` where it operates and their interactions. The examples of organisational characteristics, decision-making concepts and analysis needs are consolidated in Table 3.4.

How all aspects and characteristics of organisation, such as *CS Department* as described above, can be effectively captured and sufficiently analysed to quantitatively compare decision alternatives and choose the best option are demonstrated in the rest of this thesis. The efficacy and the effectiveness of the concept model and organisational decision-making requirements are validated using near real-life case studies in Chapter 7.

3.4 Summary

This chapter presents a foundational overview, proposes a conceptual meta-model to describe the necessary concepts, and discusses the modelling and analysis requirements for organisational decision-making. The foundational basis, conceptual meta model and the requirements are

drawn from the general system theory, cybernetics, system dynamics, complex adaptive systems, complexity theory, economics and behavioural science, and a range of management literature.

Overall, the contributions of this chapter are twofold - (i) detailed discussion on the problem space of this research and introduction of the key concepts in the form of a conceptual meta model, (ii) precise modelling and analysis requirements to enable the evidence-based quantitative organisational decision-making. The proposed conceptual meta model is considered as the concepts or vocabulary to describe organisational decision-making for the rest of this thesis whereas the modelling and analysis requirements are used for conducting literature review, define research solution and validate research outcomes.

Chapter 4

Modelling and Analysis Techniques

This chapter reviews a wide range of modelling and analysis techniques to evaluate their suitability with respect to the needs of organisational decision-making (as discussed in Chapter 3). It briefly evaluates two broad analysis categories that are considered for understanding complex enterprises or systems, namely: *qualitative approach* and *quantitative approach* [65] in section 4.1. Section 4.1 also signifies the relevance of *enterprise modelling and analysis techniques* and *actor and agent technologies* as prospective technology aids. Two comprehensive literature reviews are conducted on the state-of-the-art *enterprise modelling and analysis techniques* (henceforth *EM techniques*) and *actor and agent technologies* (henceforth actor technologies) to evaluate their capabilities with respect to the modelling and analysis needs (as presented in Table 3.3). The literature review methodology that is used is discussed in section 4.4. The review of EM techniques is discussed in section 4.3 and review of *actor technologies* is discussed in section 4.4. The chapter concludes with a synthesis of the literature reviews. It highlights the suitability and limitations of the existing modelling and analysis techniques, which defines the scope for technology improvements. From the research methodology adopted and described in Chapter 2, this chapter iterates over the research activity *Exploring the state-of-the-art of organisational decision-making* and concludes with *Define the objectives for a solution* by defining expected improvements in the state-of-the-art modelling and analysis techniques (research activities are described in Figure 2.4) .

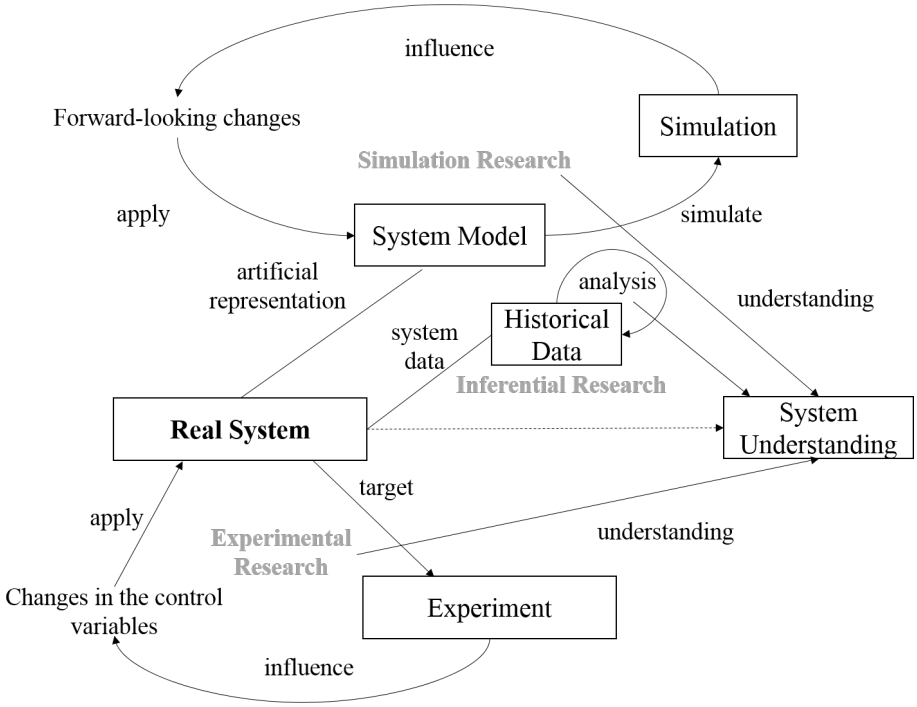


Figure 4.1 Overview of simulation research

4.1 Broad spectrum of modelling and analysis techniques

The analysis of complex systems or enterprises are typically approached using two broad analysis categories: *qualitative approach* [133] and *quantitative approach* [65]. The qualitative approach is concerned with the subjective assessment of the underlying system through a range of management techniques such as interviews, discussions, and field studies. The quantitative approach, in contrast, involves precise interpretation of system data, structure and behaviours.

The quantitative approach is further classified into three categories: *inferential approach*, *experimental approach* and *modelling and simulation approach* [113] as pictorially represented in Figure 4.1. The *inferential approach* [138] analyses the existing system data (*i.e.*, trace or historical data) to infer the characteristics of a system. The *experimental approach* comprehends a system by manipulating the system variables and observing their effects in a controlled environment. The *modelling and simulation approach*, in contrast, relies on the philosophy of *science of the artificial* [184] where one or more aspect(s) of a system is/are represented in an abstract form or a purposive model. The *modelling and simulation approach* imitates a real system using a (purposive) *model*, explores a range of scenarios by simulating the possible (forward looking) changes incorporated into the model, and develops a precise understanding about a system by interpreting the simulation results. The modelling and simulation approach is

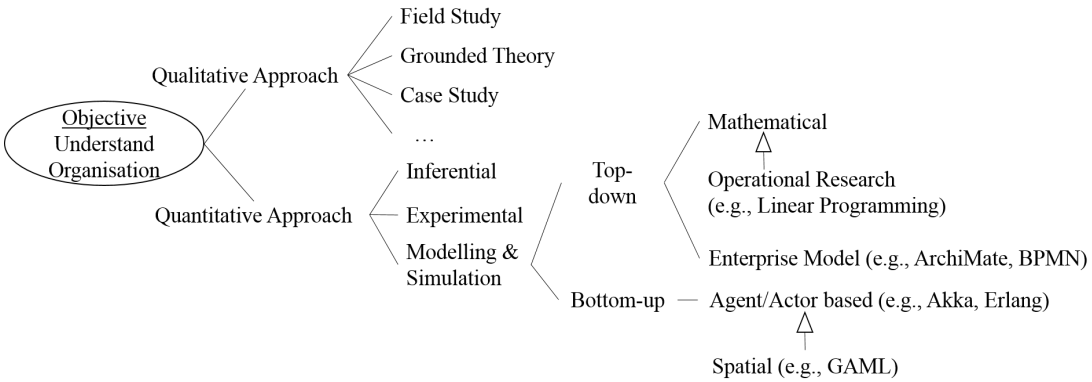


Figure 4.2 Spectrum of analysis approaches

classified into various classes and sub-classes as shown in Figure 4.2. In particular, the *modelling and simulation approach* visualises systems using two approaches: *top-down* approach and *bottom-up* approach [192]. A *top-down* approach models a system as a whole and adopts reductionist view to decompose it into smaller parts to understand the parts in isolation. This approach uses a range of models to represent and analyse the overall systems. These models are – (i) *mathematic model* and (ii) *enterprise model* (EM). The *mathematical models*, such as Monte Carlo simulation model [144], represent a system using mathematical formulae and use rigorous mathematical and statistical problem solving techniques for system analysis. The *operational research techniques* are the specialised form of mathematical models. In particular, they use models, such as linear programming [48], integer programming [176] and dynamic programming [38], for system modelling and analysis. In contrast, the *enterprise models*, such as ArchiMate [100], i* [218] and BPMN [209], and System Dynamics (SD) [134], are typically less rigorous than mathematical models, however they serve a wide range of modelling and analysis needs of the complex enterprises. A *bottom-up* approach starts from the parts or micro-behaviours and arrives at a holistic view of a system through composition. The *bottom up* approach uses the agent and actor based technologies, such as Erlang [12], Akka [5], and Scala Actor [90], for modelling and analysing systems.

This research considers *quantitative approach* as a research direction over *qualitative approach* to support quantitative *what-if* analyses. In particular, two kinds of modelling and analysis approaches: *EM techniques* and *actor technologies*, are considered from a range of analysis approaches highlighted in Figure 4.2.

The *inferential approach* relies on historical data, which works well for mechanistic systems and the class of *programmed* decision-making problems thus it is not considered for further

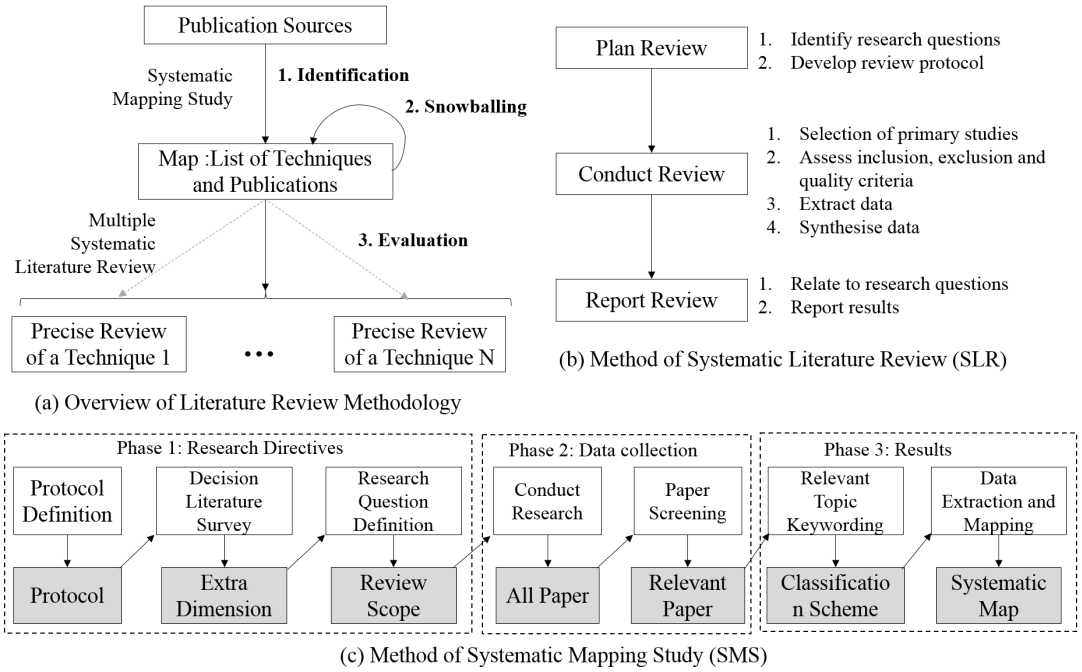


Figure 4.3 Systematic review methodology (Sources [155] and [111])

exploration. The *experimental approach* is not considered for two reasons - (i) conducting experiments on a real setting is not always an economically viable option [184, 195] and (ii) real life experiments are often restrictive and typically they are conducted in a localised context, which fail to understand the ramification of the localised changes in a global context. The modelling and simulation approach, in contrast, is less restrictive and free from historical biases as compared to the other two approaches. It helps to analyse the (hypothetical) changes and capable of observing the long term consequences under various anticipated/predicted environmental disruptions. Moreover, the simulation is considered as an effective epistemic engine to understand system when system data is not available, credibility of the existing data is questionable, conducting experiments on a real system is not a feasible option or other two options are not economically viable [184, 195]. From modelling and simulation approaches, the mathematical model is useful consideration when relevant aspects of the system can be suitably modelled using algebraic equations, which is difficult proposition for the organisations that exhibit socio-technical characteristics, significant uncertainty and emergent behaviours [183, 35]. The EM techniques are used to comprehend large and complex business enterprises [127, 172] and *actor technologies* are discussed as relevant technology to model and analyse socio-technical systems [129, 96]. Therefore, this research precisely focuses on those two

streams of modelling and analysis techniques to evaluate their suitability in organisational decision-making.

4.2 Literature review methodology

The systematic reviews to evaluate the state-of-the-art modelling and analysis techniques are performed using a two-step process as shown in Figure 4.3 (a). The objectives and the core activities of the two-process steps are summarised below:

1. **Identification**: Identification step identifies the three artifacts: (i) literature corpus for detailed reviews, (ii) the list of modelling and analysis techniques referred in the corpus, and (iii) mappings from specific modelling and analysis techniques to relevant publications. For example, the *Identification* step of the systematic review identifies (i) EM techniques related publications, (ii) a list of EM techniques such as Zachman Framework, ArchiMate, and ARIS, that are discussed in the EM publications, and (iii) the mappings from the EM techniques to publication lists (*i.e.*, a list of relevant publications for Zachman Framework, a list of relevant publications for ArchiMate, *etc*).

   Methodologically, this step adopts the process steps recommended in the conventional Systematic Mapping Study (SMS) [155] as depicted in Figure 4.3 (c). Where each SMS considers multiple digital libraries, such as *Scopus*, *ScienceDirect*, *IEEE Xplore*, and *ACM Digital Library* to ensure the literature coverage. In addition, the snowballing technique[1] as described in [212] is also considered to confirm the core publications that propose/introduce specific modelling and analysis technique is included in the review corpus. For example, the paper that introduces the Zachman Framework, *i.e.*, [220], should be included in the corpus to study the Zachman Framework.

2. **Evaluation**: The key objective of evaluation step is to evaluate the suitability of modelling and analysis techniques with respect to the modelling and analysis requirements discussed in Table 3.3. This is an iterative step where each iteration follows the process steps proposed for conventional Systematic Literature Review (SLR) [111] as depicted in Figure 4.3 (b). For example, an iteration may evaluate the suitability of Zachman Framework with respect to the requirement criteria as described in Table 3.3 using SLR methodology.

[1] Snowballing refers to use the reference list of a paper or the citations to the paper to identify additional papers

This two-steps review method is used to validate EM techniques and actor technologies. The method ensures the threat to validity of the overall review process by identifying all four kinds of *threats to validity* evident in software engineering related reviews [213] and mitigating them appropriately as discussed below:

1. Internal validity threat: Errors due to deviation from the standard or defined study protocol cause this threat. An external validation of the *review protocol* and execution steps[2], use of standard publication sources (*i.e.*, *Scopus*, *ScienceDirect*, *IEEE Xplore*, and *ACM Digital Library*) and automated publications extraction from digital libraries minimise the threats to internal validity.

2. External validity threat: Error due to inappropriate generalisation. The study of each modelling and analysis technique is considered as a different SLR and no generalisation is applied for overall review process. Therefore, the external validity is ensured in this review method.

3. Construct validity threat: This threat arises due to the constructs that characterise a study are incorrect for some reason such as inadequate publications and biases. The use of SMS prior to a SLR and *snowballing* techniques ensures literature coverage. The biases are addressed by considering multiple digital libraries.

4. Conclusion validity threats: An incorrect interpretation of the observed/extracted/experimented data due to incorrect usage of statistical methods and lack of descriptive statistics causes the threat to the conclusion validity. The use of simple descriptive statistics to interpret observed data minimises the threat to conclusion validity.

The systematic review of the EM techniques and the actor technologies using this review method are described in the next sections.

4.3 Enterprise modelling and analysis techniques

Enterprise Modelling and Analysis techniques are considered in various challenging business problems such as business-IT alignment, process improvements, enterprise transformation, and regulatory compliance [173]. This wide adoption and growing popularity of EM techniques

[2]The protocol and execution steps are reviewed by three supervisors who were not part of the review team.

Table 4.1 Review protocol for conducting systematic mapping study of EM techniques

SMS Artifact	**Artifact Description**
Research Question	RQ1: What are the papers on Enterprise Modeling (EM) and Enterprise Architecture (EA) that focus on organisation modelling? RQ2: What are the EM techniques cited by identified papers?
Inclusion Criteria	Keywords: Enterprise Architecture" OR "Enterprise Model" OR "Enterprise Modelling" OR "Enterprise Modeling" Subject Area: Computer Science Document Type: Conference and Journal Paper Language: English
Exclusion Criteria	"workflow" OR "BPR" OR "governance" OR "government" OR "security" OR "mining" OR "re-engineering" OR "Six Sigma" OR "SOA" OR "mashups" OR "Web Service" OR "Cloud" OR "data warehouse" OR "ERP" OR "SAP" OR "Digital Media" OR "MIS" OR OR "RFID" OR "sensor network" OR "network management" OR "LAN" OR "database" OR "network infrastructure" OR "NAS"
Quality Criteria	a) Publication is peer reviewed, and b) Publication is cited by at least 1 paper if publication date is prior to 2016
Sources	Scopus, ACM Digital Library, IEEE Xplore, ScienceDirect and Web of Science
Study Template	Template to capture Title, Authors information, Citation Count, EM techniques referred

in the industry practice [114, 172] make a case to explore them as a potential technology aid for organisational decision-making. This section presents an overview of the systematic review using the two-step method described in the previous section.

4.3.1 Literature identification and mapping

An SMS is conducted to identify a publication corpus and list of existing EM techniques. The overview of the SMS planning and review protocol is described in Table 4.1. As highlighted in the table, the protocol defines two broad research questions RQ1 and RQ2, an inclusion criteria, an exclusion criteria and two quality criteria.

The inclusion criterion of this review is very broad as it is designed to find all the EM and Enterprise Architecture (EA) related literature. The exclusion criterion is designed to eliminate EM techniques that solely focus on workflow, process mining, security, and infrastructure related topics as they are not relevant to organisational decision-making. Two constraints are defined as part of quality criteria. The criteria are: (i) publication is peer reviewed and (ii) publication should be cited by at least one refereed paper (excluding self-citation) if it is published before

Table 4.2 Enterprise modelling and analysis techniques

	Enterprise Modelling Techniques	**Domain**	**A-Count**	**S-Count**
1.	Zachman Framework [220]	Information System	493	23
2.	Unified Modeling Language (UML) [151]	Information System	306	14
3.	ArchiMate [100]	Information System	190	19
4.	Business Process Model and Notation (BPMN) [209]	Information System	190	17
5.	ARchitecture of Integrated Information Systems (ARIS) [175]	Information System	167	19
6.	i* [218]	Information System	71	4
7.	Multi-Perspective Enterprise Modelling (MEMO) [80]	Information System	61	10
8.	The Open Group Architecture Framework (TOGAF) [93]	Information System	53	7
9.	The Discrete Event System Specification (DEVS) [47]	Information System	47	9
10.	Business Motivation Model(BMM) [189]	Information System	38	5
11.	System Dynamics [78]	All	34	7
12.	Enterprise Knowledge Development (EKD) [168]	Information System	33	2
13.	Design and Engineering Methodology for Organizations (DEMO) [73]	Information System	28	2
14.	Event-driven process chain (EPC) [136]	Information System	23	2
15.	Petri Net [156]	Information System	22	2
16.	Semantics of Business Vocabulary and Rules (SBVR) [190]	Information System	17	2
17.	Knowledge Acquisition in automated specification (KAOS) [199]	Information System	13	1
18.	Extended Enterprise Modeling Language (EEML) [115]	Information System	9	2
19.	Reference Model of Open Distributed Processing (RM-ODP) [164]	Information System	8	2
20.	Integrated enterprise modeling (IEM) [37]	Information System	8	3
21.	European Interoperability Framework (EIF) [50]	Information System	7	2
22.	For Enterprise Modeling (4EM) [173]	Information System	5	2
23.	Systemic Enterprise Architecture Methodology (SEAM) [208]	Information System	5	1
24.	Department of Defense Architecture Framework (DoDAF) [211]	Defence	83	7
25.	Integration DEFinition (IDEF) [137]	Defence	51	6
26.	The British Ministry of Defence Architecture Framework (MoDAF) [17]	Defence	49	3
27.	Computer Integrated Manufacturing Open Systems Architecture Framework (CIMOSA)kosanke1995cimosa	Manufacturing	126	25
28.	Generalized Enterprise Reference Architecture and Methodology (GERAM) [101]	Manufacturing	92	13
29.	Graphs with Results and Actions Interrelated (GRAI) and GRAI Integrated Methodology (GIM) [51]	Manufacturing	71	17
30.	Unifed Enterprise Modeling Language (UEML)[204]	Manufacturing	43	6
31.	Purdue Enterprise Reference Framework (PERA)[210]	Manufacturing	32	5

2016. The former quality criterion checks the quality of the publication and the latter criteria[3] ensures minimum acknowledgment from the research community.

[3]Citation count is a weak quality criteria in IS research. Intentionally a weak criteria is used so as to include all existing EM techniques with a minimum filtering to exclude which are not matured or not received any acknowledgment from the research community.

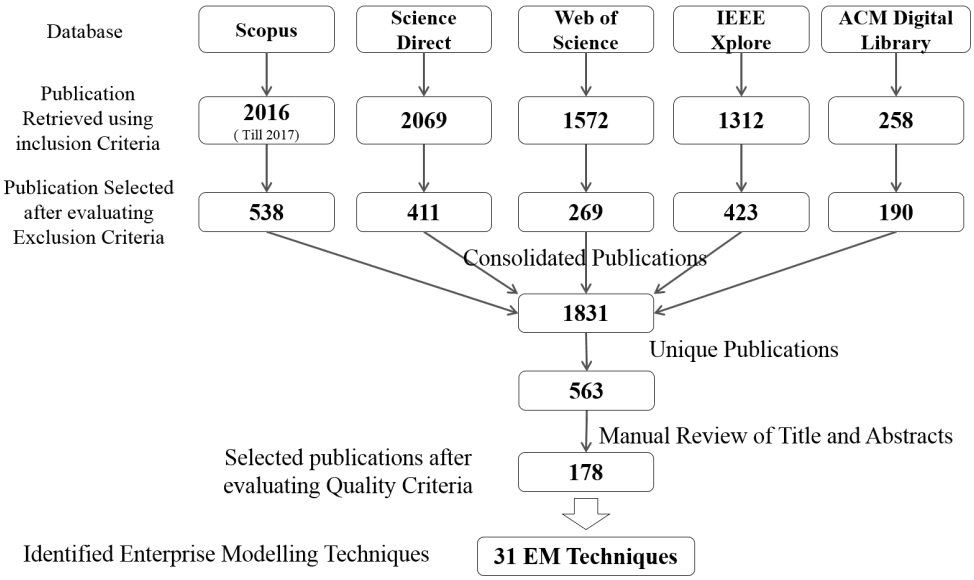


Figure 4.4 Execution of SMS on enterprise modelling and analysis techniques

The SMS conducted on the EM technique identified 178 publications from 563 unique EM and EA related publications that are collated from 5 popular digital libraries: *Scopus*, *ACM Digital Library*, *IEEE Xplore*, *ScienceDirect* and *Web of Science*. The overview of the review execution is depicted in Figure 4.4. As shown in the figure, the inclusion criteria and exclusion criteria are applied on the respective digital libraries to identify relevant publications and then they are consolidated for evaluating the quality criteria. Finally 178 publications are manually reviewed. The review concluded with 31 EM techniques as listed in Table 4.2.

As shown in the table, the EM techniques are used and exploited within three broad domains: *Information Technology*, *Manufacturing* and *Defence*. Twenty three EM techniques focus on IS. Three EM techniques, namely DoDAF, IDEF and MoDAF, are used in defence systems, and CIMOSA, PERA, GERAM, GRAI and UEML are mostly used in the manufacturing domain.

The table also presents two appearance counts. The column *A-Count* represents the appearance of a specific EM technique in 563 selected publications, and the column *S-Count* represents their appearance in 178 publications (which are considered for precise review). The former column indicates the popularity of the EM techniques. For example, the Zachman Framework, Archimate, ARIS, UML are referred very frequently in the EM literature whereas the EEML, IEM, KAOS are not referred extensively. The later column ensures the coverage in final review.

The next step of this systematic review focuses on the EM techniques from Information System (IS) domain. However, the systematic review excludes ToGAF, EIF, SEAM, SVBR and

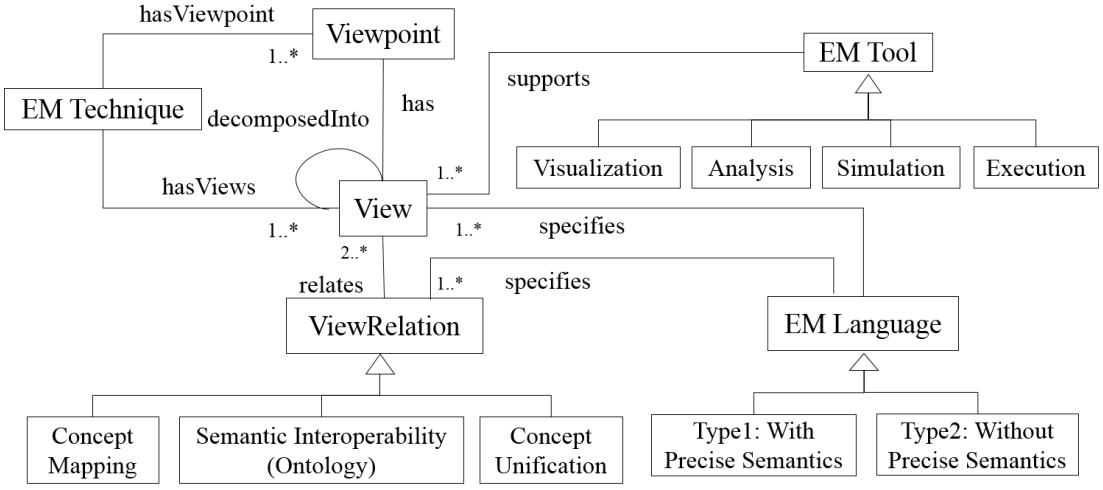


Figure 4.5 Meta model to understand EM techniques

RM-ODP for further evaluation as ToGAF and SEAM are primarily methodological guidances, the SVBR is a standard that defines the vocabulary and rules for describing the business facts and business rules, the EIF is a set of guidance to European public administrations about the design of European public services, and the RM-ODP is a reference model for the distributed systems.

4.3.2 Evaluation of EM techniques

In general, EM techniques are capable of representing or communicating a range of aspects of an organisation. They often encompasses a (set of) modelling language(s) or specification(s) to capture a set of organisational aspects, and a set of tools to visualise, analyse, simulate and/or execute the supported specification(s). The key objectives of the evaluation phase is to study the characteristics of EM techniques using iterative SLRs as depicted in Figure 4.3 (a). Each SLR iteration in this review process focuses on a specific EM technique (such as Zachman Framework or ArchiMate), review the publication list mapped to the specific EM technique, and evaluate the modelling and analysis capabilities. The key objectives of the evaluation are to evaluate:

- Capability to capture relevant aspects of organisation (as discussed in previous chapter).
- Capability to specify the organisational characteristics as discussed in Table 3.3.
- Required analyses capabilities and tool support.

	Data What	Function How	Network Where	People Who	Time When	Motivation Why
Objective/Scope:	List of Things Important in the Business	List of Core Business Processes	List of Business Locations	List of Important Organizations	List of Significant Events	List of Business Code
Business Model	Conceptual Data/Object Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
System Model	Logical Data/Class Model	System Architecture Model	Distributed Systems Architecture	Human Interface Architecture	Processing Structure	Business Role Model
Technology Model	Physical Data/Class Model	Technology Design Model	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Detailed Representations	Data Definitions	Program	Network Architecture	Security Architecture	Timing Definition	Rule Specification
Functioning Enterprise	Usable Data	Working Function	Usable Network	Functioning Organization	Implemented Schedule	Working Strategy

Figure 4.6 Overview of Zachman Framework (Source [220])

The evaluations use a template in terms of a meta-model to capture the capabilities of the EM techniques in a systematic and uniform manner. The meta-model, termed as *EM Synthesis* meta-model[4], is presented in Figure 4.5. As shown in the figure, the meta-model considers that an `Enterprise` can be visualised along multiple dimensions/perspectives, which are terms `Viewpoints`. Each viewpoint represents a set of aspects interest of an organisation, which are termed `Views`. These views can be hierarchically decomposed into a set of sub-views or more specific `Views`. A `View` may relate to another `View` through `ViewRelation`. An enterprise modelling language or specification (termed as `EM Language`) offers a set of relevant constructs to represent one or multiple `Views` and `ViewRelations`. Similarly, the `EM Tools` provide analysis and visualisation support for a set of `Views`. Possible characteristics of the `ViewRelation`, `EM Language` and `EM Tool` are described using the categorisation as illustrated in the figure. The `ViewRelation` can be described by one of the three types: `Concept Mapping`, `Semantic Interoperability` and `Concept Unification`. In `Concept Mapping`, the concepts of two `Views` of a *ViewRelation* are explicitly mapped (such as the one described in [146]). The *Semantic Interoperability* establishes the relationship between the concepts from two different `Views` using semantic mapping (as illustrated in [77]). The `Concept Unification`, in contrast, uses a unified meta-model to establish relationships between the concepts of two `Views` (as highlighted in [204]).

[4]The review template as a meta-model is used for consistency and completeness, better visualisation of the review outcomes, and produce a machine interpretable information

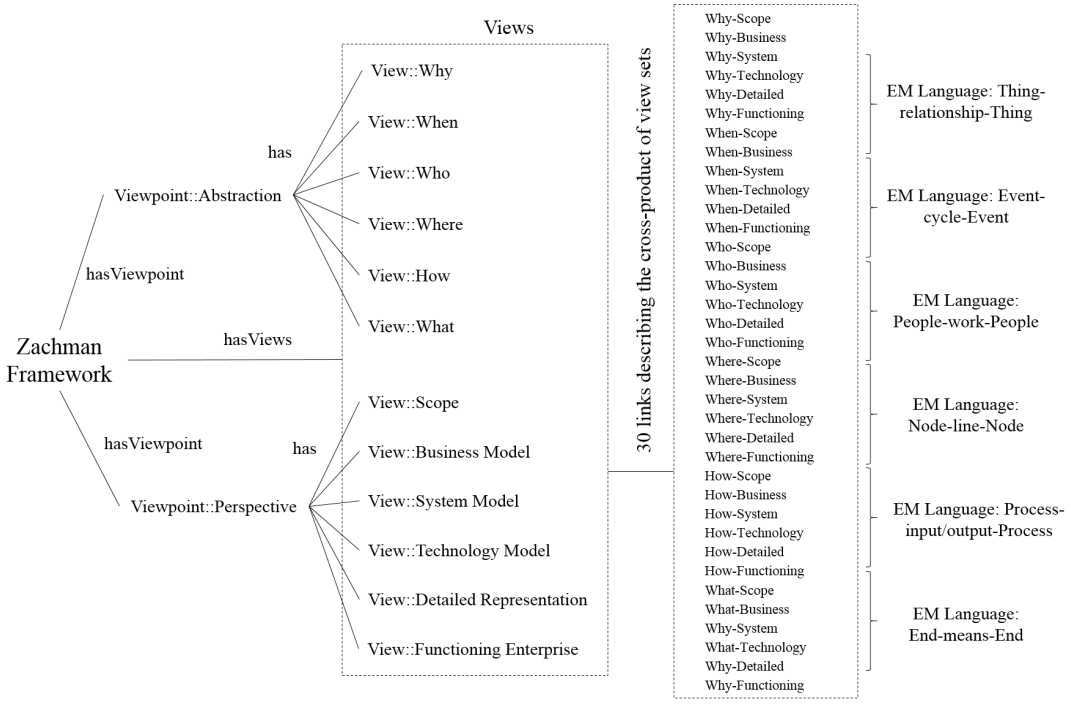


Figure 4.7 Instance model of Zachman Framework

The `EM Languages` can broadly be categorized into two types – the one with precise execution/simulation semantics (indicated as `Type1`) and the one without (indicated as `Type2`). On the other hand, the `EM Tools` can be categorized into four broad classes – `Visualization` tool, `Analysis` tool, `Simulation` tool and `Execution` tool.

The reviews of IS specific EM techniques, which are extensively referred in this thesis, are summarised in this subsection. The reviews summary of the remaining EM techniques from IS domain are discussed in Appendix A.

Zachman Framework

The Zachman Framework [220] is a structured way to visualise and define an enterprise. The framework recommends six interrogative aspects of an enterprise along six perspectives as shown in Figure 4.6. The aspects are: *What*, *How*, *When*, *Who*, *Where*, and *Why*. The *What* aspect describes the data and structure of the enterprise, *How* is the functional specification, *When* describes the time aspect, *Who* describes the people and stakeholders of the enterprise, *Where* is the description of the location, and *Why* is the description of the motivation of an enterprise. Supported perspectives are: *Scope*, *Business Model*, *System Model*, *Technology Model*, *Detailed representation* and *Functioning Enterprise*. The *Scope* is a high-level executive

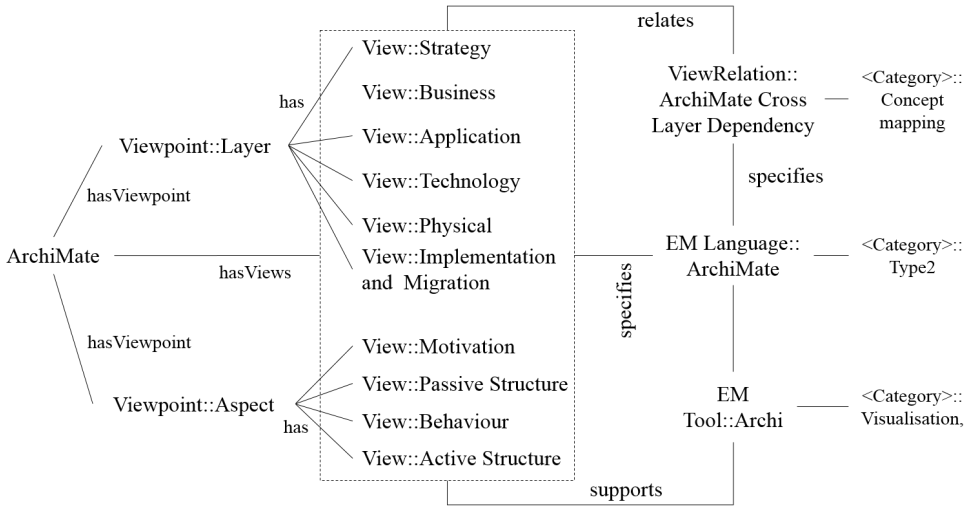


Figure 4.8 Instance model of ArchiMate

perspective, *Business Model* is the perspective of business management, *System Model* is the viewpoint of architects, *Technology Model* is an engineers or programmers perspective, *Detailed representation* is the perspective of technologists, and *Functioning Enterprise* is operational perspective of an enterprise.

This understanding derived from the SLR on Zachman Framework is represented using an instance model of the *EM Synthesis* meta-model as shown in Figure 4.7. The figure shows – the Zachman Framework supports two `Viewpoints`: *Abstraction* and *Perspective*. *Abstraction* viewpoint has six `Views`: *What*, *How*, *When*, *Who*, *Where*, and *Why*, where each *View* represents an enterprise aspect. Whereas, the *Perspective* `Viewpoint` supports six `Views` namely: *Scope*, *Business Model*, *System Model*, *Technology Model*, *Detailed representation* and *Functioning Enterprise*. The Zachman framework has thirty six `ViewRelations` and supports six kinds of specifications or `EM languages` as shown in the figure. The supported specification types are: *Thing-Relationship-Thing*, *Event-cycle-Event*, *People-work-People*, *Node-line-Node*, *Process-input/output-Process* and *End-means-End*. The `ViewRelations` are `Concept Mapping` type relations, and the EM languages are `Type1` type, *i.e.*, specification without any execution/simulation semantics.

ArchiMate

ArchiMate [100] is one of the most popular enterprise modelling language that supports the description, analysis and visualization of the organizational structures, business processes, IT systems, technical infrastructure and information flows of an enterprise in an unambiguous

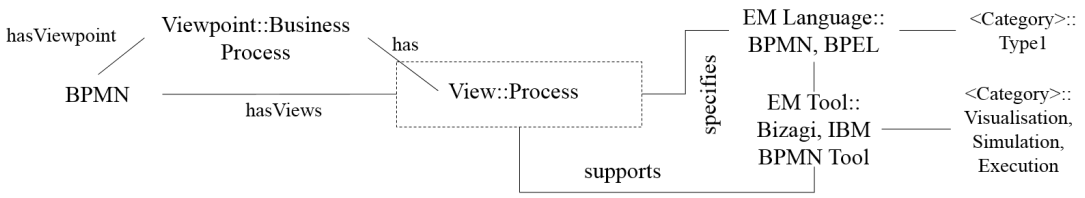


Figure 4.9 Instance model of BPMN

manner. The core ArchiMate language describes an enterprise along three layers: *Business Layer*, *Application Layer* and *Technology Layer*. The concepts of each layers are categorised into three aspects: *Active Structure Aspect*, *Behaviour Aspect* and *Passive Structure Aspect*. The complete ArchiMate language supports additional three layers (namely: *Strategy Layer*, *Physical Layer* and *Implementation and Migration Elements*) and a aspect (namely *Motivation Aspect*).

The *Strategy Layer* describes the business strategies, *Business Layer* specifies the business services offered to the customers, *Application Layer* depicts the application services that support the business, *Technology Layer* depicts the technology infrastructure, such as processing, storage, and communication services, *Physical Layer* describes physical elements such as equipment, facilities, and distribution network, and *Implementation and Migration Elements* describes the work package, deliverables, operational constraints, *etc.*

The *Active Structure Aspect* represents the structural elements that have their own behaviour, such as business actors, application components, and devices; the *Behavior Aspect* represents the behaviour, such as processes, functions, events, and services; and the *Passive Structure Aspect* represents the information objects, and the *Motivation Aspect* describes the motivations that includes *value*, *meaning*, *driver*, *assessment*, *goal*, *outcome*, *etc.*

The instance model produced from the SLR on ArchiMate is depicted in Figure 4.8. As shown in the figure, the ArchiMate visualises an enterprise along two `Viewpoints`: *Layer* and *Aspect*. The *Layer* `Viewpoint` has six `Views`, and the *Aspect* `Viewpoint` has four `Views` respectively. The `Views` are related with each other using *Cross Layer dependency* mappings, the `EM Language` called as *ArchiMate* is `Type2` category specification, the the popular `EM Tool` known as *Archi* [5] is a visualisation tool.

[5]https://www.archimatetool.com/

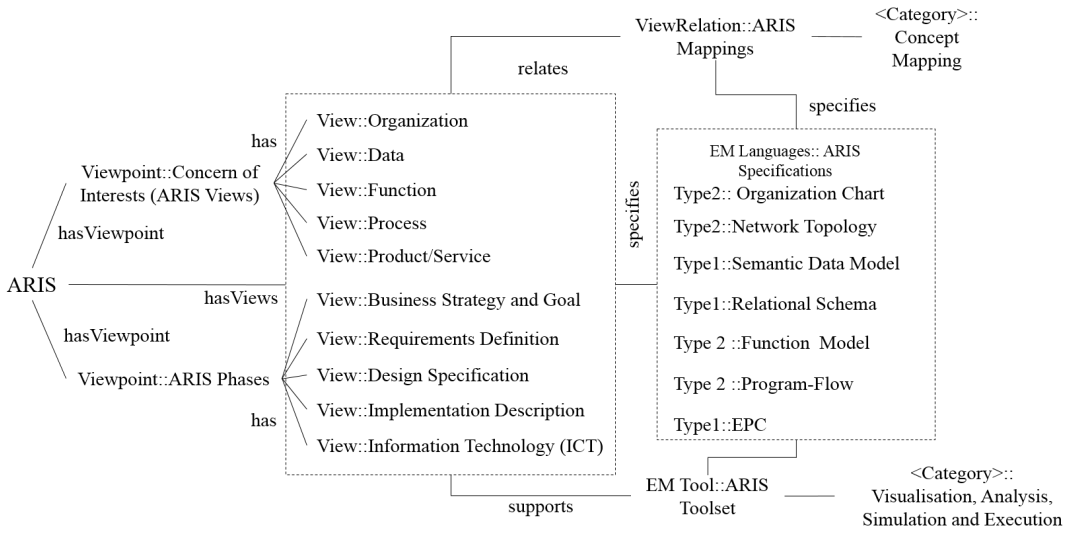


Figure 4.10 Instance model of ARIS

Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) [209] is a language and notation for specifying the business processes using flowcharting technique, which is similar to the *activity diagram* of the Unified Modelling Language (UML).

The outcome of the SLR on BPMN is shown in Figure 4.9. The BPMN supports single `Viewpoint` and a `View` that focuses on the business process aspect of an organisation. The BPMN specification is a `Type2` EM language. However, the Business Process Execution Language (BPEL) is a `Type1` EM language, *i.e.* it has a precise simulation and execution semantics. The BPMN tools, such as Enterprise Architect[6], Bizagi[7], IBM Business Process Management (BPM) tool[8], offer a wide range of visualisation, simulation and execution capabilities.

ARchitecture of Integrated Information System (ARIS)

ARchitecture of Integrated Information Systems (ARIS) [175] is a multi-perspective enterprise modeling approach that supports the modelling and analysis of various aspects of an enterprise, and offers sophisticated industry-scale tools to simulate and execute the business process specification. It helps to capture five views of an enterprise: *Organisational Structure*, *Data*, *Process*, *Functions* and *Product*.

[6] http://www.sparxsystems.com/products/ea/trial/request.html
[7] https://www.bizagi.com/en
[8] https://www.ibm.com/us-en/marketplace/business-process-manager

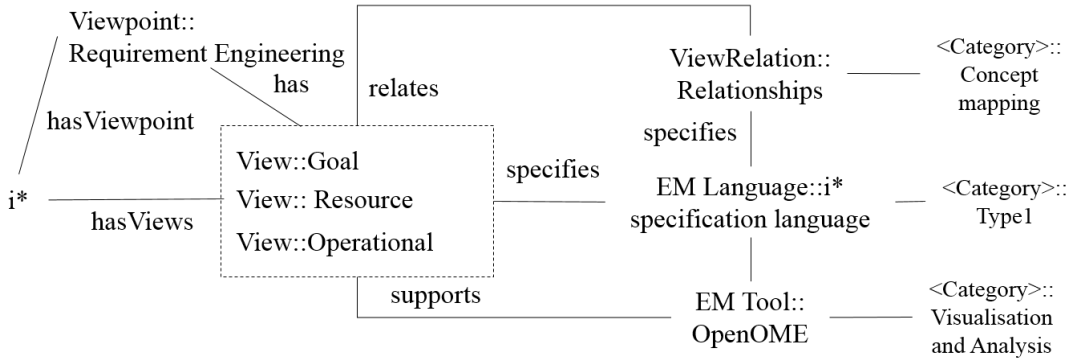


Figure 4.11 Instance model of i*

The instance model produced from SLR on ARIS is depicted in Figure 4.10. The ARIS supports two `Viewpoints`: *ARIS Views* and *ARIS Phases*. Both the `Viewpoints` have five `Views` respectively as shown in the figure. The `Views` are related to each other through `ViewRelations` of category `Concept Mapping`. The ARIS supports a range of `EM Languages` with varying semantics categories. Supported `EM Languages` are: Organization Chart (`Type2` category), Network Topology (`Type2` category), Semantic Data Model (`Type1` category), Relational Schema (`Type1` category), Function Model (`Type2` category), Program Flow (`Type2` category) and Event-driven Process Chain (EPC) (`Type1` category). The `EM Tools`, such as *ARIS Architect*[9], supports a varying range of capabilities that include visualisation, analysis, simulation and execution.

i*

The language i* [218] is an agent-oriented modelling language for requirement engineering that represents the goals, beliefs, abilities, and commitments of an enterprise and their strategic relationships.

It supports the `Viewpoint` of *Requirement Engineering* and three `Views` that describe the *Goal*, *Resource* and the *Ability* of an enterprise as shown in Figure 4.11. The i* specification language has a unified meta-model to describe all three `Views` and has a precise semantics (*i.e.*, `Type1 EM Language`). The i* specification is supported by a range of sophisticated visualisation and analysis tools[10].

[9] https://www.softwareag.com/corporate/products/aris_alfabet/bpa/aris_architect/default
[10] http://www.cs.toronto.edu/km/istar/Software

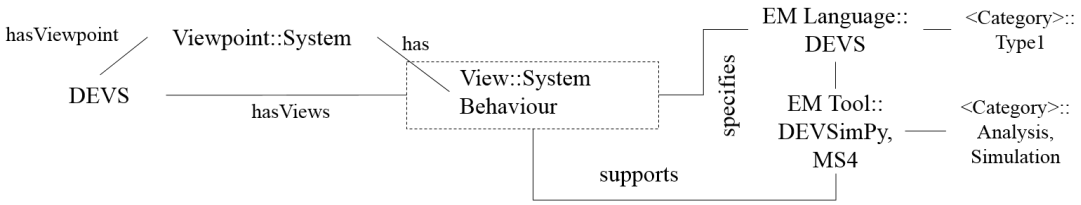


Figure 4.12 Instance model of DEVS

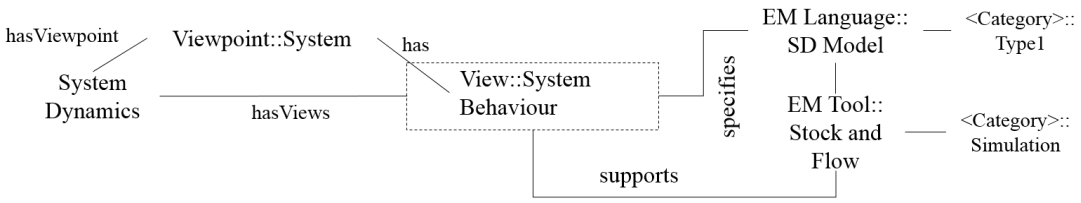


Figure 4.13 Instance model of System Dynamics

Discrete Event System Specification (DEVS)

Discrete Event System Specification (DEVS) [47] is a modular and hierarchical modelling language to model and analyse the system behaviour using discrete event formalism. The specification is suitable for deterministic and causal systems. It allows the behavioural specification at two levels. At the lowest level, an atomic DEVS describes the autonomous behaviour of the fine grained units of a system, and at the higher level, a coupled DEVS describes a system as a network of coupled components. The components of a coupled DEVS can be atomic DEVS models and/or coupled DEVS models.

The model produced from the SLR on DEVS is depicted in Figure 4.12. It supports the *System* `Viewpoint` and *System Behaviour* `View`. The DEVS modelling language is a `Type1` EM Language and DEVS-based tools, such as DEVSimPy[11] and MS4[12], support sophisticated analysis and discrete event simulation.

System Dynamics (SD)

The System Dynamics (SD) [78] is a modeling technique that helps to understand the nonlinear behaviour of a complex system over time. An SD model describes system behaviour using the concepts of *stocks*, *flows*, *auxiliary variables*, *feedback loops*, *table functions* and *time delays*, where the dynamism is specified using differential equations over time. The model produced by applying the SLR on the SD approach is depicted in Figure 4.13. As shown in the figure,

[11] https://github.com/capocchi/DEVSimPy
[12] http://www.ms4systems.com/pages/devsjava.php

Table 4.3 Review synthesis of EM techniques

EM Technique	Specification Capabilities						Organisational Characteristics								Concepts			Analysis Aids				
	Why	What	How	Who	When	Where	Modularity	Composability	Reactive	Autonomous	Intentional	Adaptable	Uncertainty	Temporal	Goal	Measure	Lever	Spec Type	Emergentism	Visualisation	Analysis	Simulation
Zachman	S	S	S	S	S	S	S	N	N	N	N	N	N	N	I	I	N	T2	N	N	N	N
UML	I	S	S	S	I	I	S	S	N	N	N	N	N	N	I	S	I	T1	N	S	S	N
ArchiMate	S	S	S	S	I	I	S	I	S	N	S	N	N	N	S	S	I	T2	N	S	S	N
BPMN	N	I	S	S	I	I	S	S	S	N	N	N	N	N	N	S	N	T1	N	SHow	SHow	SHow
ARIS	I	S	S	S	I	I	S	S	S	N	N	N	N	N	I	S	N	T1	N	SHow	SHow	SHow
i*	S	N	N	I	N	N	S	S	N	N	N	N	N	N	S	S	I	T1	N	SWhy	SWhy	SWhy
MEMO	S	S	S	S	N	N	S	S	I	I	N	N	N	I	I	S	N	T1	N	S	S	I
DEVS	N	N	S	N	S	N	S	S	S	N	N	N	N	N	N	N	N	T1	N	SHow	SHow	SHow
BMM	S	N	I	N	N	N	I	N	N	N	S	N	N	N	S	I	N	T2	N	N	N	N
SD	N	I	S	N	I	N	I	N	I	S	N	N	I	S	N	N	N	T1	N	I	S	SHow
EKD& 4EM	S	S	S	S	N	N	S	S	N	N	N	N	N	N	I	I	N	T2	N	S	I	N
DEMO	N	I	S	I	N	N	I	I	S	N	N	N	N	N	N	N	N	T2	N	SHow	SHow	N
EPC	N	I	S	N	I	S	S	S	S	N	N	N	I	S	N	N	N	T1	N	SHow	SHow	SHow
Petri Net	N	I	S	N	S	S	S	S	S	N	N	I	S	N	N	N	N	T1	N	SHow	SHow	SHow
KAOS	S	I	N	I	N	N	S	S	N	N	N	N	N	N	S	I	N	T1	N	I	I	N
EEML	I	S	S	S	N	N	N	N	N	N	N	N	N	N	S	I	N	T2	N	I	N	N
S=Suitable, Sx = Suitable for Aspect X, I=Inadequate, N=Not Suitable, T1=Type1, T2=Type2																						

the SD technique is capable of representing the *System Behaviour* `View` using SD model or Stock-and-Flow model. The SD models are `Type1 EM Language` and the `EM Tools`, such as iThink[13] and Simantics[14], support system dynamics simulation.

4.3.3 Review report of EM technologies

The SLR iterations evaluated the EM techniques with respect to the desired modelling and analysis needs. The evaluations are summarised in Table 4.3. As shown in the table, the EM techniques exhibit a wide range of modelling and analysis capabilities. The capabilities of the EM techniques represented in Table 4.3 are analysed along three dimensions – (i) modelling capability (*i.e.*, as presented in `Specification Capabilities` columns and `Organisational characteristics` columns of Table 4.3), (iii) decision-making concept specification capability (*i.e.*, data from `Concepts` columns), and (iv) analysis capabilities (*i.e.*, data from `Analysis Aids` columns). The findings are discussed below:

[13]https://www.iseesystems.com/
[14]http://sysdyn.simantics.org/

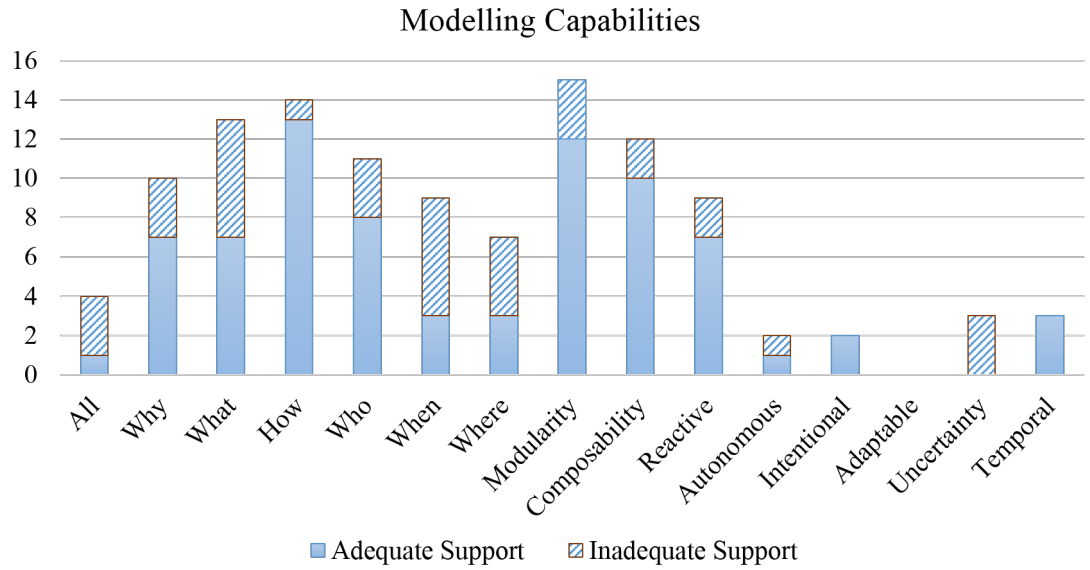


Figure 4.14 Modelling capabilities of EM techniques

1. **Aspect specification capability**: Table 4.3 shows a consensus among EM techniques that an enterprise should be specified along multiple views or aspects. It is visible throughout the table as all the EM techniques support more than one aspects. However, only four EM techniques (*i.e.*, Zachman Framework, UML, ArchiMate, and ARIS) out of 16 EM techniques are capable of specifying all the six aspects: *Why*, *What*, *How*, *Who*, *Where* and *When*. Moreover, the support for specifying the characteristics of the complex organisation is a major concern across EM techniques.

   The modelling capabilities of the EM techniques are pictorially shown in Figure 4.14. As shown in the figure, the EM techniques are mostly process-oriented as 14 EM techniques out of 16 support the *How* aspect of an enterprise. They are *modular* and *composable*; not all EM techniques support *reactiveness*; and not suitable for specifying *adaptability*, *uncertainty*, *intentionality*, *autonomous* behaviour and *temporal* characteristics.

2. **Decision-making concept specification capability**: As shown in the table, five EM techniques: ArchiMate, i*, BMM, KAOS and EEML support adequate constructs to specify the *Goals* and *Measures*. On the other hand, the *Goals* and *Measures* can also be specified using UML, Zachman, ARIS, MEMO and 4EM through some indirect means. For example, the *Goals* can be specified using UML by introducing a new stereotype. However, none of the EM technique is capable of specifying the *Lever* of an organisational decision making problem.

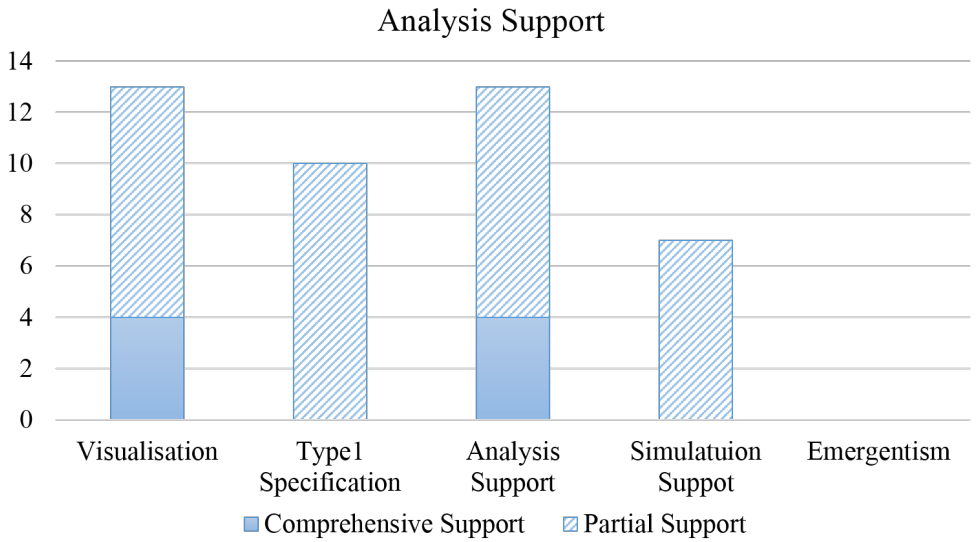


Figure 4.15 Analysis capabilities of EM techniques

3. **Analysis capability**: The analysis capabilities of the EM techniques are shown in Figure 4.15. As shown in figure, they mostly support the visualisation as an aid to understand an enterprise or organisation. Thirteen EM techniques out of 16 support visualisation. However, there is a considerable lacunae in `Type1` EM language – 10 EM techniques offer *Type1* specification but none of them is capable of specifying all six aspects of an enterprise. Similarly, the support for simulation is also limited to a specific aspect. The review also identifies that none of the EM technique is capable of describing and observing the emergent behaviour.

The above analysis leads to a conclusion that the state-of-the-art enterprise modeling and analysis techniques are not adequate to address the needs in a comprehensive manner. The key inadequacies are three-fold:

1. Inability to capture the inherent characteristics of the organisation that include *autonomous* behaviour, *adaptability*, and *uncertainty* (as shown in respective columns of Table 4.3).

2. Lack of language constructs to specify decision-making concepts such as *Lever* and *Measure* (*see* `Concepts` section of Table 4.3).

3. Inability to model and observe the emergent behaviour of a system (as it can be seen in `Emergentism` column of Table 4.3).

Table 4.4 Review protocol for conducting SMS on actor technology

SMS Artifact	**Artifact Description**
Research Question	RQ1: What are the publications on actor and agent technologies? RQ2: What are the actor/agent technologies cited by identified papers?
Inclusion Criteria	Keywords: "Actor Language" OR "Actor Framework" OR "Actor Computation" OR "Actor Model" OR "Agent Language" Document Type: Conference and Journal Paper Language: English
Exclusion Criteria	"animation" OR ("multiprocessor" AND "architecture") OR "hypervideo" OR "hypermedia" OR ("virtual" AND "reality") OR "SDH Network" OR "embedded" OR "mobile" OR "wireless" OR "sensor" OR "video" OR "movie" OR "health" OR "medical" OR "Strategic Actor"
Quality Criteria	a) Publication is peer reviewed, and b) Publication is cited by at least 1 paper if publication date is prior to 2016
Sources	Scopus, ACM Digital Library, IEEE Xplore, and ScienceDirect
Study Template	Template to capture Title, Authors information, Citation Count, Actor technology referred

4.4 Actor and agent technologies

Actor and *agent* based systems are capable of representing, analysing and implementing complex systems using the notion of an *actor* or an *agent*. Conceptually, an *agent* is an *autonomous*, *self-contained*, *reactive*, and *pro-active* entity that can communicate with other *agents* through message passing [216]. Similarly, *actor* [96] is a primitive and universal concept for a range of distributed and concurrent computations. In the *actor model of computation* [2], *actors* are computation units that support modularity, autonomy and reactive behaviour. This section reviews the *actor* and *agent* technologies to evaluate their suitability. The systematic review is conducted using a two-step process as shown in Figure 4.3 (a). The brief overview of the review, review outcomes and the review summarisation are discussed in this section.

4.4.1 Literature identification and mapping

A Systematic Mapping Study (SMS) using a review protocol described in Table 4.4 is conducted to identify the relevant literature on actor and agent technologies. As shown in the table, the inclusion criterion is broad to include frameworks, languages and models on *actors* and *agents*. The exclusion criterion is designed to eliminate languages and frameworks for video, streaming, embedded systems, sensors, wireless devices, and mobile systems. The study also excludes the actor/agent technologies, which are associated with the medical science, and the notion of

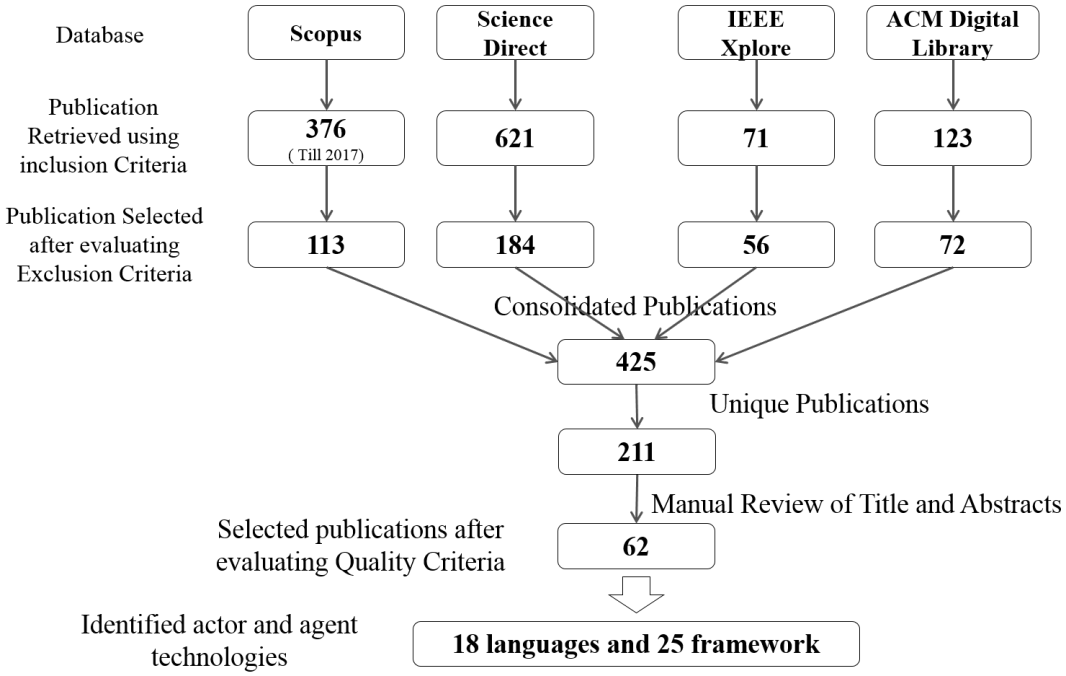


Figure 4.16 Execution of SMS on actor technologies

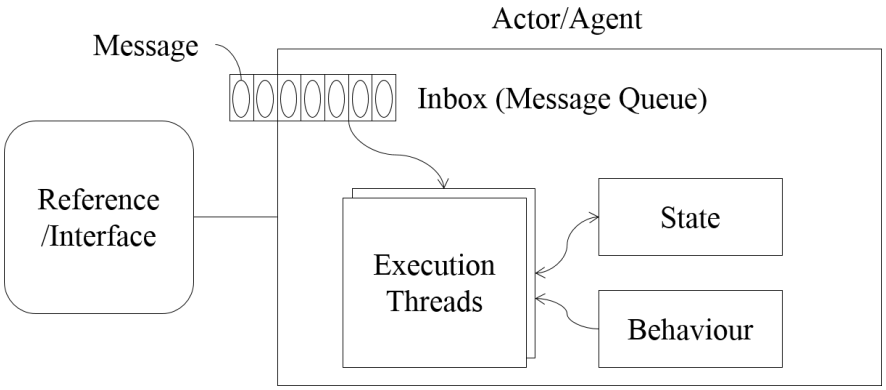


Figure 4.17 Conceptual overview of an actor or agent

actors from i* specification as they are not relevant for this study. Two constraints are defined as part of quality criteria as discussed in the previous study that focuses on EM techniques.

The overview of the review execution is depicted in Figure 4.16. As shown in the figure, the mapping study identifies 62 publications from 211 unique actor/agent technology related publications that are collated from four digital libraries: *Scopus*, *ACM Digital Library*, *IEEE Xplore*, and *ScienceDirect*. The review recognises two kinds of technologies - (i) actor/agent languages and (ii) actor/agent frameworks and libraries. The *languages* offer a specification language and its runtime system, whereas the frameworks offer a set of APIs on a specific

Table 4.5 Actor technologies

Actor Languages		
1. Erlang [13], 2. Akka [5], 3. BDI [163], 4. SALSA [202], 5. Act [3], 6. E [140], 7. Actor Based Concurrent Language (ABCL) [217], 8. Rebeca [186], 9. Act 1, 2 and 3 [125], 10. Rosette [196], 11. ThAL [109],12. PLASMA [97], 13. Harmony/2[207], 14. Pony [59], 15. SARL [167], 16. GAML [88], 17. AgentSpeak [162], 18. 3APL [98]		
Actor frameworks		
	Framework	Underlying Language
1.	Scala Actor [90]	Scala
2.	AnyLogic [44]	Scala
3.	Kilim [187]	Java
4.	AmbientTalk [72]	Java
5.	Act++ [104]	C++
6.	Broadway [188]	C++
7.	Stackless Python [193]	Python
8.	Acttalk [46]	Smalltalk
9.	Actor Foundry [15]	Java
10.	Stage [16]	Ruby
11.	Actor Architecture [103]	Java
12.	Jetlang [165]	.NET
13.	JavAct [11]	Java
14.	AJ [221]	Java
15.	NetLogo [194]	-
16.	JADE [36]	Java
17.	JADEX [159]	Java
18.	Repast [61]	Java
19.	Karus [https://code.google.com/archive/p/korus]	Java
20.	Pykka [https://pypi.python.org/pypi/Pykka]	Python
21.	Nact [https://code.google.com/archive/p/n-act]	.NET
22.	Microsoft Asynchronous Agents Library [https://msdn.microsoft.com/en-us/library/dd492627(VS.100).aspx]	C++
23.	Jetlang [https://github.com/jetlang]	Java
24.	actor-cpp [https://code.google.com/archive/p/actor-cpp]	C++
25.	Orleans [http://dotnet.github.io/orleans]	.NET

technological platform to specify, analyse and simulate *actors* or *agents*. The review recognises 18 actor/agent languages and 25 frameworks as listed in Table 4.5.

4.4.2 Evaluation of actor and agent technologies

The capabilities of actor and agent technologies listed in Table 4.5 are evaluated using a series of SLRs as discussed in Section 4.2. Conceptually, these technologies support a set of common characteristics. An encapsulated entity named as *actor* or *agent* represents distributed and interacting units of a system. As shown in Figure 4.17, each *actor* or *agent* (henceforth *actor*) has an *interface* as an identity, an *inbox* or message queue for communication or interactions,

and an encapsulated and self-contained computing unit. The computing unit encapsulates *actor's* state, behaviour and a set of execution threads. Each *actor* has a thread which can update its state. The threads are concurrent and cannot share actor state. The messages are asynchronous and fair [2], however the order of the message delivery is not guaranteed in an actor communication.

Based on the purpose of the computation, the actor and agent technologies are broadly categorised into two families: (i) languages and frameworks for general purpose distributed and concurrent computing and (ii) distributed and concurrent computing to capture and understand the *spatial* influences such as cellular automata [214] or *game of life* [63]. The languages GAma Modeling Language (GAML) and NetLogo, Java Agent DEvelopment Framework (JADE), JADEX, and Repast frameworks from the list depicted in Table 4.5 solely focus on the *spatial* relationships. They are better suited to study the social and political systems, which is beyond the scope of this thesis.

The review summary of the selected general purpose actor/agent languages and frameworks (henceforth actor technologies) that are referred extensively in the literature are discussed below:

1. Actor languages

- **Actor**: *Actor* language proposed by Gul Agha [2] realises the notion of *actors* as a set of concurrent objects. Each of these actors interacts with each other through asynchronous message passing. The language supports three language primitives – `create`, `send` and `become` to specify an actor system and its behaviour. The primitive `create` creates an actor from a behaviour description and returns the address of the newly created actor; the primitive `send` asynchronously sends a message and immediately returns the control to the sender; and `become` replaces the behaviour of an actor.

- **Erlang**: Erlang [12] is a declarative general-purpose industry-strength actor-based language. It realises the notion of an *actor* using lightweight *processes*. The Erlang processes communicate using message passing. Each process has a *mailbox* that stores the messages, which are received but not processed. The Erlang *processes* use sophisticated pattern-matching to identify the next message to be processed; the message-handling logic processes a message if pattern is matched; and finally the messages are removed from the mailbox when message is processed.

- **E**: The language E [140] realises the notion of an *actor* using a concept termed as *vat*. A *vat* has an event queue, a heap of objects, a stack and single thread of control.

The *vat* encapsulates the objects that it owns and sends object references to other *vat* using messages passing. When *a* vat receives a message, it enqueues the messages and immediately returns a *promise* as an envelope. The *promise* is eventually resolved with the return value of the message once that message is processed.

- **ABCL**: Actor-Based Concurrent Language (ABCL) [217] is an object-oriented concurrent programming language. It realises the notion of an *actor* using the concept of an active *object*. Each *object* has its own local persistent memory and a thread of control. In ABCL, the state changes are not specified in terms of behaviour updates (*i.e.*, `become`) instead it uses assignment statements. It supports three types of messages passing – `past`, `now` and `future`. The control of a `past` type message is immediately returned to the sender. The `now` type messages are similar to the synchronous function calls. The `future` type messages return a reference to query the return value in future.

- **BDI**: Belief-Desire-Intention (BDI) [163] is an agent-based programming language that captures the *beliefs*, *desires* and *intentions* of an agent, and tries to achieve the *desires* by using the *beliefs* and *intentions*. In BDI, the *beliefs* of an agent are the information or facts that exist within an agent, the *desires* are the motivational state, and *intentions* represent the actions and plans. It uses sophisticated inference rules and forward chaining for developing new beliefs, and relies on temporal logic and Computation Tree Logic (CTL) for reasoning and problem solving.

2. Actor frameworks

- **Akka**: Akka [5] is an industry-scale actor framework or a library developed using Java and Scala platform that runs on JVM. In Akka, the actors can be modeled as Java/Scala objects where each actor is identified using a reference, which is known as `ActorRef`; an actor contains a mailbox to hold incoming messages until they are processed, and actors communicate with others through asynchronous messages. Akka supports a hierarchical actor structure where an actor may create multiple child actors. A parent actor supervises its children by delegating activities which the child can handle and handling the exceptions, which the child cannot handle. The Akka actor cannot enforce pure actor encapsulation as the underlying Java/Scala object allows mutable data structures and capable of sending references to other actors/objects.

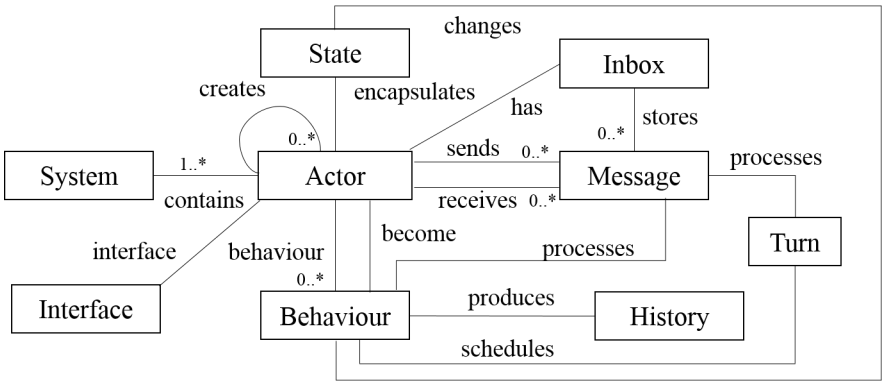


Figure 4.18 Topology of an actor based system

- **Scala Actor**: The *Scala Actor* [90] is a library that offers a full-fledged implementation of Erlang-style actors on top of Scala. The Scala actors are concurrent processes that communicate through exchanging messages. It supports both asynchronous and synchronous message sends. In addition, the actors may communicate using *futures*. The messages are asynchronous but they return *future* as an reference to query the return values.

- **AnyLogic**: AnyLogic [44] is an agent-based simulation technology. It considers a concept of an *agent* to specify an organisation using a set of modular units. It uses discrete event for message passing between the agents, and adopts a multi-model and co-simulation approach for agent specification and simulation. The multi-modelling and co-simulation includes the formalism such as Stock and Flow diagram [78], Statechart [91], Action Chart [44] and Process Flow Charts [44].

4.4.3 Review report of actor and agent technologies

The SLR iterations evaluate a range of actor and agent technologies that consider *actor* and *agent* as a universal concept and propose a common set of constructs to describe an actor/agent-based system. A generic schema of the concepts proposed in the actor and agent languages is depicted in Figure 4.18. As shown in the figure, an actor based `System` (that represents both the term, *i.e.*, *actor* and *agent*) is typically composed of a set of *modular*, *self-contained*, and *interactive* `Actors` (it is also termed as *active object*, *agent*, *activity*, *vat* and *grain* in various actor/agent languages/literature). Each `Actor` has its `Interface`, `State`, `History`, `Inbox` (`Inbox` is also termed as *mailbox*, *message queue* and *event queue*) and a set of autonomous `Behaviour`. An `Actor` interacts with each other through sending and receiving `Messages` (the

Table 4.6 The capabilities of actor and agent technologies

	Requirements	**Support**	**Discussion and Exception**
Specification Capabilities	Why	Supported only in BDI technology	No explicit construct to capture *Why* aspect in rest of the actor and agent technologies
	What	Support simple structure	1. Most of the actor and agent technologies support simple actor structure 2. Akka supports a hierarchical actor structure
	How	Supported	Constructs `Behaviour`, `Turn` and `Message` specify the *How* aspect
	Who	Supported	*Who* aspect can be specified using `Actor`
	When	Not supported	No guarantee in the order in which the `Messages` can be delivered makes the *When* computation difficult for an actor `System`
	Where	Supported	*Where* aspect can be represented using `Actor` of an actor `System`
Organisational Characteristics	Modularity	Supported	Inherent characteristic of actor and agent technologies
	Composability	Limited support	The actor and agent frameworks rely on the underlying languages for composition, whereas the the actor and agent languages mostly support the distributed `Actors`
	Reactive	Supported	Sending and receiving `Messages` help to specify the reactive nature
	Autonomous	Supported	Inherent characteristic of all actor and agent technologies
	Intentional	Supported only in BDI technology	No other actor and agent technology is capable of specifying the intention of an `Actor`
	Adaptable	Supported	Constructs `new` and `become` help to specify the adaptability
	Uncertainty	Not supported	No construct to capture uncertainty in an explicit manner
	Temporality	Not supported	No construct to represent temporal behaviour in an explicit form
Concepts	Goal	Supported only in BDI technology	No other actor and agent language is capable of specifying the *Goal*
	Measure	Not supported	No support in any of the actor and agent technology
	Lever	Not supported	No support in any of the actor and agent technology
Analysis Aids	Top-down & Bottom-up analysis	Partially Supported	The bottom-up analysis is supported in most of the actor and agent technologies
	Emergentism	Supported	Actor and agent technologies are capable of producing emergent behaviour
	What-if Analysis	Partially Supported	Actor and agent technologies support bottom-up simulation

`Message` is also known as *envelop*, *event*, and *request*). The `History` and `State` of an `Actor` are encapsulated and they can only be accessed through message passing. The `Behaviour` of an `Actor` is responsible for producing `History`, `State` change, message passing, and new `Actor` creation in an actor `System`. An `Actor` can change its `Behaviour` and replace it with another using a construct/API, which is typically termed as `become`. The `Behaviour` of an `Actor` schedules `Turn` (also known as *epoc* and *step*) that processes the `Messages` from the `Inbox`.

The `Actors` in most of the actor and agent technologies are typically characterised by *modularity*, *encapsulation*, *reactiveness* and *autonomous* behaviour. They are capable of changing a system topology by creating new `Actors` and have ability to change the `Behaviour`

of existing `Actors` using the `become` construct. Therefore, they are capable of specifying the desired adaptability of an actor `System`. The autonomous `Behaviour` and unrestricted message passing further help to produce the *emergent* behaviour of an overall actor `System`. However, the existing actor and agent technologies are not capable of describing the intention of an `Actor` in an explicit form. They are also not suitable to specify the inherent uncertainty and temporal `Behaviour`. The consolidated evaluation of the actor and agent technologies with respect to the requirements described in Table 3.3 (of Chapter 3) is presented in Table 4.6.

As shown in the table, the state-of-the-art actor and agent technologies support the desired modularity, reactiveness, autonomy, and adaptability. They are capable of representing and observing the emergent behaviour. However, they lack the following desired characteristics:

1. Lack of support for expressing complex structure. Moreover, it is hard to represent the *Why* and *When* aspects using an actor language/framework.

2. Lack of language constructs to specify decision-making concepts, such as *Goal*, *Measure* and *Lever*.

3. Existing actor languages and frameworks do not natively support *uncertainty* and *temporal* behaviour.

4.5 Synthesis of literature reviews

The literature reviews on the state-of-the-art modelling and analysis techniques report a wide spectrum of modelling, analysis and simulation capabilities. At one extreme are mathematical models, such as linear programming, integer programming and dynamic programming, that represent systems using mathematical equations and use mathematic techniques, such as optimisation, for precise analyses and problem solving. However, their use is largely limited for deterministic and bounded systems. They are best suited for *programmed* decision-making problems. The other class of models are various EMs. From the spectrum of enterprise models, a class of enterprise models provide a well-defined structure to represent the organisational aspects and offer a variety of visualisation techniques to help humans obtain the desired understanding of the organisation. For instance, ArchiMate is one such specification. The other class of enterprise models are machine interpretable and/or simulatable specifications. They are capable of precise analyses for one or limited aspects. For instance, BPMN analyses and simulates the behavioural aspect, i* analyses the high level goals and objectives, and SD model simulates

dynamic behaviour of the system. The multi-modelling and co-simulation environments, such as DEVS and MEMO, demonstrate further advancements by supporting the analysis of multiple aspects. Principally, they adopt a top-down approach to model an organisation and use a reductionist view for precise understanding. They are not capable of formulating a suitable environment to construct and observe the inherent emergentism of an organisation.

The languages and specifications advocating the *actor model of computation* and the agent-based systems support emergentism through bottom-up simulation. They fare better in analysing the systems with socio-technical characteristics such as modularity, autonomy, reactiveness and adaptability. However, they do not support the specification of complex goals, organisational hierarchies, and behavioural uncertainty. Moreover, all kinds of models fall short as an intuitive and closer-to-the-problem specification as they are not designed for organisational decision-making. In particular they lack the concepts goal, measure and lever.

From a methodological viewpoint, the goal specification languages such as i*, EKD and KAOS advocate a top-down method. The EM techniques such as ArchiMate, MEMO, and 4EM also advocate a top-down method and a globalized view of the system to represent the goal, structure and behaviour of an organisation in an integrated manner. BPMN and SD model predominantly support a top-down approach and reductionist view for a range of analyses. The actor and agent languages and frameworks, in contrast, advocate the localised specification and bottom-up analysis approach.

The other methodological advances follow the similar trend. For example, the DEsign Specification of Interacting REasoning components (DESIRE) [201] and MEMO based decision-making process [40] propose top-down modelling method and *what-if* analysis based on reductionist viewpoint. The methodology advocated in [110] supports the bottom-up approach using BDI paradigm. Principally, none is capable of combining top-down/bottom-up design principles, reductionist/emergentism analysis techniques, composional/decompositional abstractions, and localized/globalized perspectives as desired for organisational decision making.

Therefore, it can be argued that these techniques capture only a fragment of what ought to be captured and analysed for an effective organisational decision making as illustrated the requirements described in Table 3.3. From the spectrum of modelling and analysis capabilities reviewed in this chapter, the *actor* model and actor-based simulation techniques are found as the closest match towards modelling and analysis needs for organisational decision making. This research considers the *actor* model as a principal abstraction to represent complex organisation

and uses actor-based simulation technique as an analysis aid by advancing the state-of-the-art actor technology as discussed in the next chapter.

4.6 Summary

This chapter reviews the existing modelling and analysis techniques using SMS and SLR methodologies with an aim to evaluate their suitability with respect to the modelling and analysis needs highlighted in Table 3.3. The reviews show the capabilities and limitations of the modelling and analysis techniques, and highlight the utility of the *actor model of computation* and actor based simulation. The reviews provide a detailed view of *what* can be adopted and *why* they can be adopted in this research. The next chapter discusses the research contributions that adopt *actor model of computation* and actor based simulation as a conceptual approach and propose extensions to existing technologies to overcome the limitations highlighted in this chapter.

The key contributions of this chapter are – the detailed reviews on enterprise modelling techniques and actor technologies as no review on any of the two topics exists in IS literature, precise gap analysis of the existing modelling and analysis capabilities, and a research direction to model complex organisation and approach organisational decision-making.

Chapter 5

An Actor-based Simulation Aid

This chapter presents a pragmatic modelling and analysis approach along with a method to address organisational decision-making through quantitative *what-if* analysis. The proposed approach introduces three research contributions: (i) a domain specific specification, named as *OrgML*, to represent an organisation and the organisational decision-making problem in an intuitive and machine interpretable form (*i.e.*, *Contribution 2* of this thesis) (ii) an approach to translate the OrgML specification into simulation specification (*i.e.*, *Contribution 3*) and (iii) a method to construct model, validate it and perform *what-if* analyses (*i.e.*, *Contribution 4*).

From DSR perspective, the OrgML is introduced as a *Model* artifact of the *Constructs* presented in Chapter 3, and the proposed approach to translate OrgML specifications into simulation specifications and the presented method are realised as *Method* artifacts to effectively utilise the research contributions. Methodologically, this chapter focuses on *Conceptualization of proposed solution* research activity of the research method considered for this research (as shown in Figure 2.4).

This chapter is organised into three parts – solution considerations, background information and solution. The philosophical, conceptual, methodological and technological standpoints adopted in the proposed approach are discussed as the solution considerations in section 5.1. The conceptual and technological foundations that are used while conceptualising research contributions are discussed as the background information in Section 5.2. A detailed descriptions of the proposed approach and research contributions are presented in the rest of this chapter. An overview of the proposed approach is presented in section 5.3. The proposed OrgML meta-model is presented in section 5.4. The OrgML model to ESL specification transformation

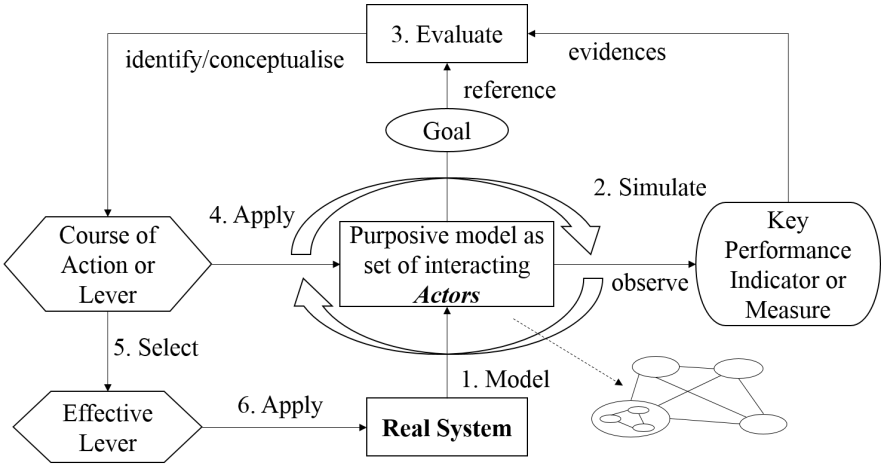


Figure 5.1 Research consideration of proposed solution

strategy and rules are discussed in section 5.5, and a detail description of the proposed method is presented in section 5.6.

5.1 Solution considerations

A schematic overview of the research direction towards the proposed solution and the design considerations are highlighted in Figure 5.1. As shown in the figure, the solution adopts a model driven approach to represent an organisation, relies on simulation techniques to produce quantitative evidence, and a human-in-the-loop evaluation step to evaluate the produced evidence. The proposed solution considers an iterative loop where the courses of action can be introduced as model change and their consequence can be observed by simulating the changed model to decide most suitable course of action that can be implemented to the real organisation. The key philosophical, conceptual, methodological and technological standpoints considered in the proposed approach are:

1. Use of philosophical and methodological foundations of *modelling and simulation* to conceptualise the overall approach. The foundations of the proposed approach can be traced to *science of the artificial* [184], *conceptual modelling* [195] and methodological viewpoint of simulation [174] as highlighted below:

   (a) The philosophical viewpoint of the *science of the artificial* [184] is used to represent the organisation as purposive *models* as shown in Figure 5.1.

   (b) *Simulation* [195] as an epistemic engine for organisational decision-making.

(c) The modelling methodology proposed by Stewart Robinson [166] and Andreas Tolk et al. [195] to construct models from a real context.

(d) The simulation method proposed by Robert Sargent in [174] to develop a simulation environment.

2. Consideration of *actor model of computation* [2] as a conceptual abstraction to model complex organisation as highlighted in Figure 5.1.

3. Use of the management viewpoint proposed by Richard Daft in [70] to represent the decision problem and capture purposive model of an organisation in a systematic manner.

4. Use of ESL, which is conceptualised and developed as part of overarching research agenda as discussed in the Introduction chapter, as an underlying simulation technology.

The primary reasons for adopting *modelling and simulation* approach are – it is open-ended (*i.e.*, any hypothetical change can be modelled and analysed), free from historical biases (as data is generated from simulation), and capable of analysing the systems in quantitative terms. The concept of *actor* is adopted in this research as *actors* are modular, composable, autonomous and reactive entities, which collectively help to imitate the real systems/organisation and it helps to observe the emergent behaviour of the organisation as discussed in section 4.4. The organisational decision-making method proposed by Richard Daft brings a management perspective into the proposed solution. The ESL technology is chosen over the existing actor languages, such as Erlang [12], Scala Actor [90] and Akka [5], due to its extensions that include the support for *uncertainty*, the notion of *time* and restricted variable sharing as discussed in section 5.2.2. However, this research does not disregard the use of existing actor languages as alternative simulation means as discussed later of this chapter.

5.2 Background

As indicated in the previous section, the proposed solution is principally based on four foundations that include – (i) modelling and simulation approach, (ii) actor model, (iii) management viewpoint of organisational decision-making method, and (iv) actor technology (in particular the ESL technology). The concept of *actor* is discussed in Chapter 4 and the management perspective of the organisation decision-making is discussed in Chapter 3. The necessary background information to introduce the proposed solution for remaining two foundational aspects

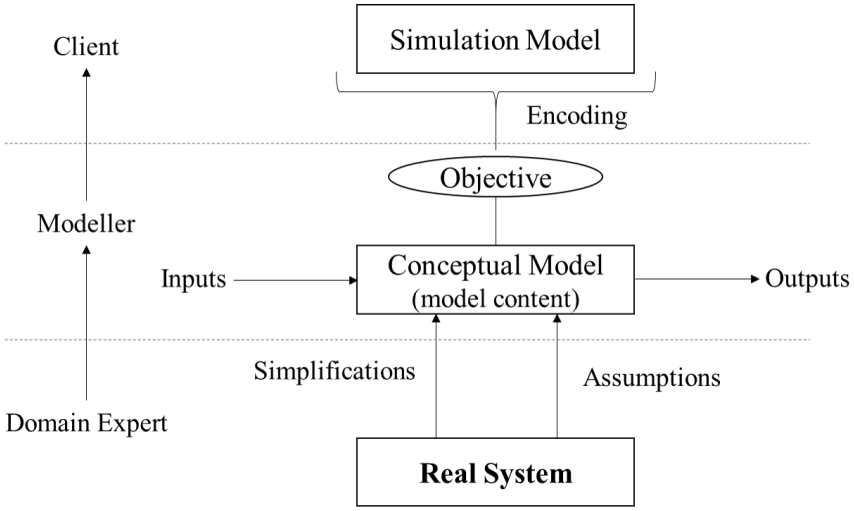


Figure 5.2 Modelling architecture for simulation research

are discussed in this section. In particular, a brief overview of the philosophical, conceptual and epistemological perspective of the modelling and simulation approach is discussed in section 5.2.1, and an overview of ESL language, which is considered as an underlying specification and simulation engine in the proposed solution, is introduced in section 5.2.2.

5.2.1 Modelling and simulation

The proposed approach constructs a purposive and faithful model of the real organisation for necessary *what-if* analyses. In this context, the constructed model provides an environment to introduce various hypothetical changes (*i.e.*, levers), and simulation of the constructed model with/without change helps to understand the trends of the key performance indicators (or measures) over time through observation of simulated results. Multiple such iterations with various changes on (base) model and observations of key performance indicators help in developing the knowledge about as-is (to-be) organisation and the possible consequences of the courses of action.

Model synthesis as an epistemic tool for system understanding is an accepted technique. In engineering fields, the scientific models that represent some aspects or parts of the real systems are extensively used for system understanding as discussed by Mieke Boon *et. al* in [43]. The enterprise modelling that represent aspects ranging from semantic to pragmatic viewpoints is also a well established discipline for understanding complex systems and organisations [172]. Simulation based is recognised as an important analytical tool in many disciplines [81]. However, the epistemic value of the knowledge developed through simulation of a model is

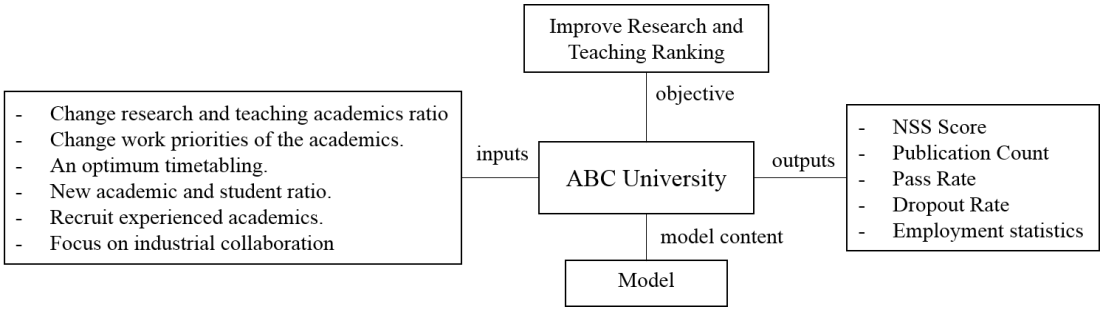


Figure 5.3 Illustration of conceptual model

largely determined by the faithfulness of the models (*i.e.*, how well a model represents the relevant aspects of a real system) [82]. The simulationists deploy a methodological rigour to ascertain the faithfulness of the constructed models[1] as discussed in [195]. This section describes a modelling architecture and a validation method that are used extensively in simulation research.

Modelling architecture

Computer-aided simulation recommends a two-layer modelling architecture that comprises the *conceptual model* and *computer model* to develop a machine interpretable model from a real system [166, 195]. Reflecting on the cognitive process of the modellers and domain experts, Stewart Robinson [166] and Andreas Tolk et al. [195] introduce an additional abstraction layer in the modelling architecture. They consider the conceptual model as a mental model and introduce a new abstraction called as *model design* to represent the concrete description of the conceptual model. This research, however, considers the canonical two-layer modelling architecture[2] as shown in Figure 5.2. The core concepts of *conceptual model* and *computer model* are described below:

- **Conceptual model**: A conceptual model (or a model design) is a purpose specific view of the real system. A conceptual model principally captures four aspects: `objectives`, `inputs`, *outputs*, and `model content` [166]. The `objectives` represent the purpose of the conceptual model. The `inputs`, or experimental factors, are the modelling elements that can be altered to represent a problem situation. `Outputs` or responses are the

[1] The model is first constructed, it is then instantiated with known system data, and finally the constructed model is simulated for several known scenarios to ensure the veracity of the constructed model and simulation tool.

[2] This modelling architecture that has close resemblance to Model Driven Architecture (MDA) architecture [112] where the *conceptual model* is similar to Platform Independent Model (PIM) that captures the necessary information using a domain-specific platform independent form and *computer model* is a Platform Specific Model (PIM) as it has a semantic to simulate on a specific platform.

modelling elements that demonstrate the reachability of the modelling objectives. The `model content` captures the abstract representation of a real system.

For an example, a conceptual model of ABC university (which is discussed in Section 1.6 of Chapter 1) is depicted in Figure 5.3 for illustration purposes. As shown in the figure, the `objective` of the conceptual model is to analyse how to '*Improve Research and Teaching Ranking*' of ABC university, the experimental factors or `inputs` that are considered – '*Change research and teaching academics ratio*', '*Change work priorities of the academics*', '*An optimum timetabling*', '*New academic and student ratio*', '*Recruit experienced academics*', and '*Focus on industrial collaboration'*. The `outputs` that needs to be observed to ascertain the reachability of specified objectives are – '*NSS Score*', '*Publication Count*', '*Pass Rate*', '*Dropout Rate*', and '*Employment statistics*'.

Two techniques, *i.e.*, *assumptions* and *simplifications*, are used to abstract a *model content* from a real system [166]. The *assumption* is way to incorporate the uncertainties, unknown factors, and the beliefs about the real system. For example, an academic may focus on any of the activities from a list of alternatives[3] at a given time. Specifying such behaviour using a probabilistic distribution is an *assumption*. Specifying the propensity of a student to raise a query or complaint in a given situation is another example of an assumption. The *simplification* is a technique to reduce the complexity of the reality by ignoring certain aspects of the real system. The purposive nature of the model and modelling scope help to achieve simplification [195]. The academics and students may get involved in a wide range of non-academic activities from various aspects such as health, recreation and entertainment and social work. Ignoring those activities in a model or considering all those activities as one *representative* activity with a propensity factor while modelling academic and student is an example of *simplification*.

- **Computer model**: A computer model is a machine interpretable form of the captured conceptual model. It is typically a mathematical or software model. For example, System Dynamics (SD) [78] uses a set of differential equations to represent a whole system in an aggregated form, and considers the concept of *stock* and *flow* to derive the subsequent state of a complex system. The agent-based and actor-based simulations [129, 5, 12] use

[3] Such as Research, Paper Writing, Writing Research Grand Proposal, Project Work, Research Collaboration, Query Resolution, Complain Resolution, Prepare for Lecture, Delivering Lecture, Prepare for Student Assessment, Student Assessment, Project Work as discussed in Table 1.1

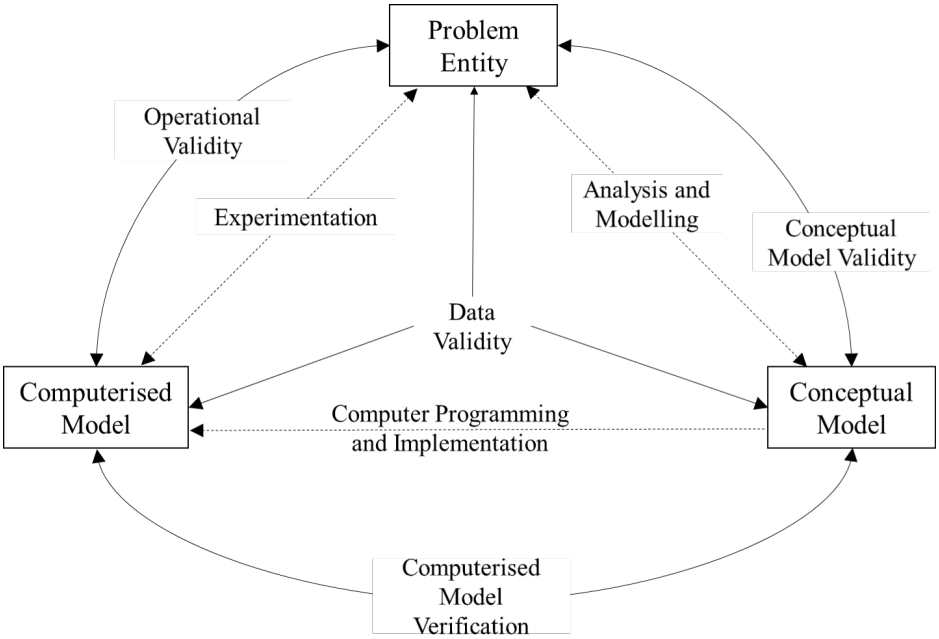


Figure 5.4 Modelling and validation method proposed by Robert Sargent [174]

the concept of interacting *agents* or *actors* to represent a system, and rely on emergentism [150] to predict the output of a model.

Simulation method

A simulation method proposed by Robert Sargent in [174] is used extensively in simulation research. As shown in the Figure 5.4, the simulation method recommends three representations: *problem entity*, *conceptual model* and *computerised model*. The problem entity is the real system or the knowledge about the system (*refer* Figure 4.1). The conceptual model is a purpose specific view of the problem entity and the computerised model is a simulatable form of the developed conceptual model as discussed in section 5.2.1.

In this method, a conceptual model is constructed from a problem entity during the *Analysis and modeling* phase, and a computerised model is encoded from conceptual model in the *Computer programming and implementation* phase. The synthesis on the problem entity is conducted by simulating/executing the computerised model in *Experimentation* phase. Each phase recommends a set of validations as shown in Figure 5.4. The *Conceptual model validity* ensures the *assumptions* and *simplifications* are *reasonable*, *i.e.* the intended purpose is sufficiently captured in the conceptual model and the underlying assumptions of the conceptual model are correct. The key consideration of this validity is to capture complete and accurate

information of the problem entity as discussed by Nelson *et al.* in [147], Krogstie *et al.* in [116], and Moody *et al.* in [143]. The *Computerised model verification* ensures the faithful translation of the conceptual model into *Computerised model*. The *Operational validity* ascertains that the simulation results are sufficiently accurate, whereas the *Data validity* ensures the data necessary for model construction, model validation, and model simulation are adequate and correct. The validation techniques that are extensively considered in the simulation study are:

- *Operational Graphics and Animation*: This technique verifies a model by observing simulation results through graphs and animations.
- *Data Comparison*: Establish validity of a model by comparing simulation results with respect to the simulation results of a valid model. For example, comparison of a simulation model with an analytical model, comparison of a simulation model that is constructed using Stock-and-Flow with a valid simulation model that is constructed using linear programming, *etc*.
- *Historical Data Validation*: Compare simulation results with the historical data of the real system.
- *Traces*: Capture behaviour of the model entities in the form of *traces* and analyse them through appropriate visualisation or analytical techniques.
- *Sensitivity Analysis*: Conduct sensitivity analyses of the input and internal parameters of a the model to determine the effect upon the output of the model. The relationships should be consistent with the known relationships of the real system.

5.2.2 Enterprise Simulation Language (ESL)

A general purpose actor technology named as Enterprise Simulation Language (ESL) to specify and simulate the complex enterprises is developed as part of the overarching research initiative[4] of this research (as discussed in Chapter 1). ESL supports the notion of concurrent actors, asynchronous and fair message passing, and the notion of history that encapsulates the actor specific execution traces as recommended in *Actor* definition [96].

In order to specify complex enterprises, ESL extends the traditional *Actor* definition along three dimensions: (i) breaking of pure encapsulation of the state space of an actor, *i.e.* an actor

[4] http://tonyclark.github.io/ESL/index.html

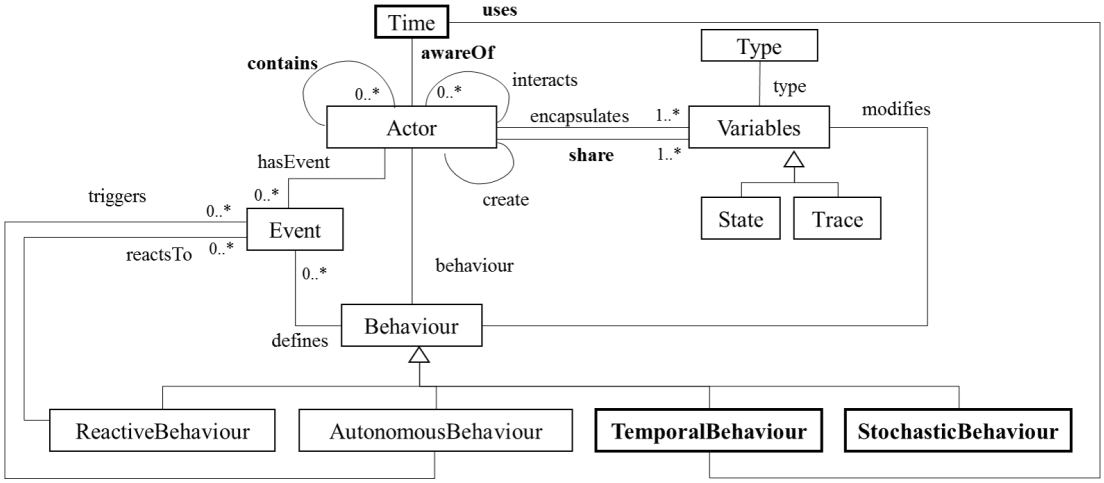


Figure 5.5 ESL meta-model

can access the exposed variables of other actors in read only mode, (ii) explicit support for the stochastic behaviour to support the inherent uncertainties, *i.e.*, support for specific construct to specify non-deterministic behaviour and (iii) support for the notion of *Time*, *i.e.*, support for relative time and its progression.

The schematic representation of ESL is depicted using a meta-model in Figure 5.5. The extended concepts are highlighted with bold text and boxes with the thick border. As shown in the figure, the concept `Actor` interacts with other `Actors` through `Events`. An `Actor` encapsulates and shares a set of typed `Variables`. These `Variables` represent the `State` and `Trace` of the `Actors`. Structurally, an `Actor` may contains a set of `Actors` and each `Actor` can create new set of `Actors`. The `Actors` are cognizant of `Time` and they have their own `Behaviour`. The `Behaviour` of an `Actor` principally represents four kinds of behavioural patterns that include reactive behaviour, autonomous behaviour, temporal behaviour and stochastic behaviour. The ESL meta-model represents these behavioural patterns using four kinds of Behaviours: `ReactiveBehaviour`, `AutonomousBehaviour`, `TemporalBehaviour` and `StochasticBehaviour` respectively.

ESL supports the standard language constructs such as assignment, expression evaluation, loop, message passing, and creation of new actor to express standard `Behaviour`. In addition, the `StochasticBehaviour` can be expressed using `probably(p) x y` construct that evaluates to `x` in p% of cases and otherwise to `y`. The `ReactiveBehaviour` reacts to an `Event` or a set of `Events`, the `AutonomousBehaviour` is typically triggered based on the state `Variables`, and the `TemporalBehaviour` is specified as an expression over `Time`.

```
0  /* Legend: ESL Keywords */
1  export main;
2  type Student = Act {Time(Int),QueryResolutionSession() };
3  type Academic = Act{StudentQuery(Student,Int)};
4
5  act student(p_name::Str)::Student {
6   export studentName;
7   studentName::Str = p_name
8   QueryResolutionSession → { ... };
9   Time(t::Int)→ { probably(20) { academic1 ← StudentQuery(self,t) } else {}}
10 };
11
12 act academic(p_name::Str)::Academic {
13  export studentsWhoRaisedQueries;
14  academicName::Str = p_name;
15  studentsWhoRaisedQueries::[Str[Int[Str]]] = []
16
17  StudentQuery(originator::Student, time::Int) → {
18   probably(80) {
19    studentsWhoRaisedQueries:= studentsWhoRaisedQueries +  [[originator.studentName,time,'
       Attended']];
20    originator ←  QueryResolutionSession
21   } else {
22    studentsWhoRaisedQueries:= studentsWhoRaisedQueries + [[originator.studentName,time,'
       NotAttended']]
23   }
24  }
25 };
26
27 academic1::Academic = new academic('Professor');
28 student1::Student = new student('Student1');
29 student2::Student = new student('Student2');
30 simulationTime::Int = 20;
31
32 act main::Main {
33  Time(time::Int) when time > simulationTime → {
34   print(academic1.studentsWhoRaisedQueries);
35   topAll()
36  };
37  Time(time::Int) → {}
38 }
```

```
[[Student1,1,Attended],[Student1,4,Attended],[Student1,8,NotAttended],[Student2,8,Attended
    ],[Student2,12,Attended]]
```

Figure 5.6 ESL specification

As an illustrative example, an ESL specification that introduces two actor types: *Academic* and *Student* is shown in Figure 5.6. The *Student* actor type consumes `Time` and `QueryResolutionSession` (shown in line 2), wherein the `Time` event is a primitive `Time` event that is internally raised by the ESL simulation engine. The *Academic* actor type consumes `StudentQuery` event as shown in line 3.

The illustrative definitions of *Student* and *Academic* actor types are shown line numbers 5–10 and 12–25 respectively. As listed, the *Student* has an actor `Variable` named as *studentName* (line 7), implements *QueryResolutionSession* event (line 8), subscribes and specifies the behaviour for `Time` event (behaviour is defined in line 9) and exports *studentName* (as specified in line 6). The behaviour for `Time` event illustrates an uncertain scenario. The specification (described in line 9) states that a *Student* actor may raise *StudentQuery* in every `Time` event and the probability of raising a *StudentQuery* for a student is 20%.

The *Academic* actor type contains two `Variables` – *academicName* and *studentWhoRaisedQuery* (as shown in line 14 and 15) wherein the *studentWhoRaisedQuery* `Variable` captures

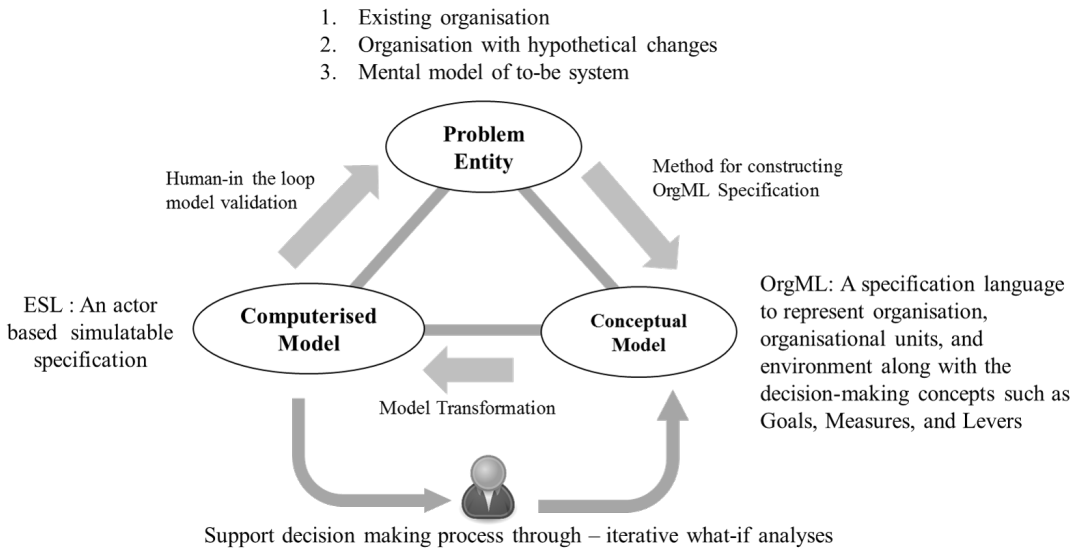


Figure 5.7 Overview of proposed approach

the trace of the *StudentQuery* related information, *i.e.* who has raised the query, when it is raised, and what is the response:– 'Attended' or 'NotAttended'. Definition of *StudentQuery* event (line 17–26) states that an academics respond to *StudentQuery* by raising (arranging) a *QueryResolutionSession* for 80 % cases (line 18–21) otherwise they choose not to respond. However, both the cases the academic actor captures the trace of *StudentQuery*' along with the response as shown in line 19 and 22 respectively. While capturing the *StudentQuery* related information, the *academic* actor accesses an exposed `Variable` of *student* actor named as *studentName*.

The rest of the ESL specification shows the actor instantiations and definition of '*main*' actor that indicates the start of a simulation run. In particular, the specification instantiates one *Academic* actor (line 27), two *Student* actors (line 28 and 29), sets simulation time (line 30) and specifies '*main*' actor (line 32 – 38). The '*main*' actor eventually prints the trace '*studentWho-RaisedQuery*' (line 34) and stops all actors at the end of a simulation run. A simulation outcome of the specification described above is also shown in bottom of the Figure 5.6.

5.3 Overview of proposed solution

An overview of the proposed solution is depicted in Figure 5.7. As shown in the figure, the proposed solution conceptually considers three abstraction layers: *Problem Entity*, *Conceptual*

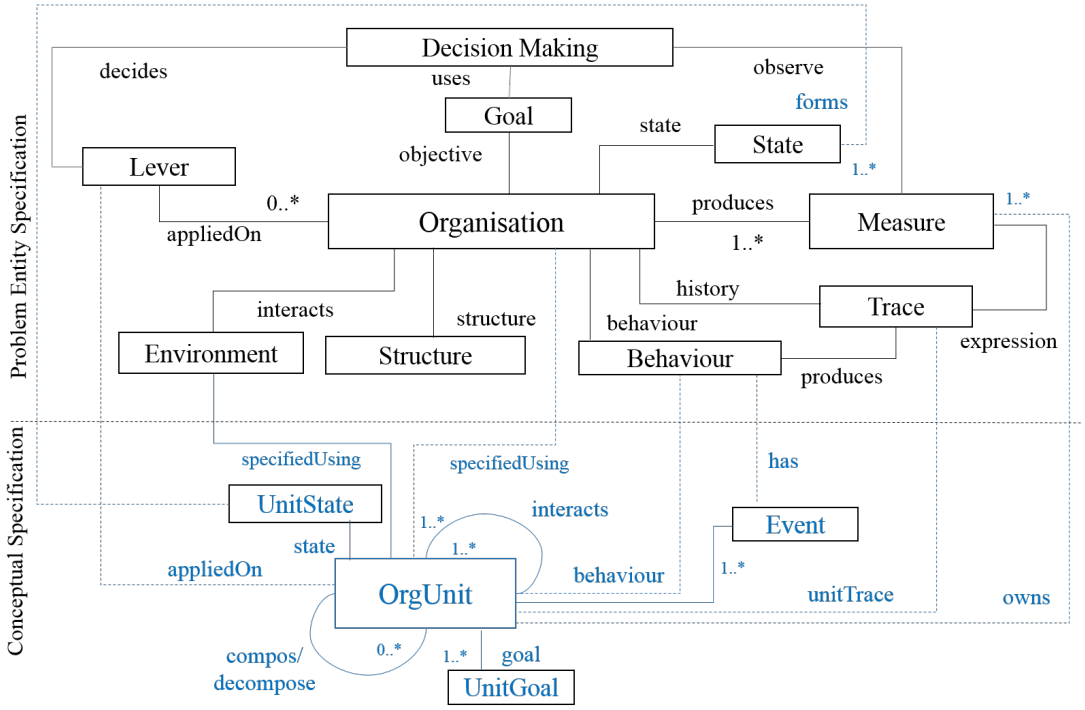


Figure 5.8 Realisation of Problem Entity using Conceptual Model

Model and *Simulation Model* as recommended by Robert Sargent in [174]. The key aspects that are considered for each of these abstraction layer are described below:

1. **Problem Entity**: *Problem Entity* represents the real organisation, organisation with hypothetical changes (*i.e.*, courses of action or levers), or a mental model of the to-be organisation as highlighted in Figure 5.7. The necessary information about the organisational aspects derived from the management literature[5] forms the *Problem Entity*. In reality, the Problem Entity information can be found in spreadsheets, pictorial representations, software models, and/or in the form of tacit knowledge of the domain experts.

2. **Conceptual Model**: The *Conceptual Model* is a purposive and machine-interpretable representation of the *Problem Entity*. The conceptual model has the following characteristics:

   (a) Conformance to the *Conceptual Model* presented by Stewart Robinson [166] and Andreas Tolk et al. [195] as depicted in Figure 5.2. The model captures *objective*, *inputs*, *outputs* and *model content*.

[5] As depicted in Figure 3.6 of section 3.3.1 in chapter 3 and also shown as *Problem Entity Specification* in Figure 5.8

(b) Captures the necessary aspects of organisation (as depicted in Figure 3.6 and highlighted as *Problem Entity Specification* in Figure 5.8) using recursive decomposition in the form of a specialised *actor*, termed as `OrgUnit`.

A conceptual schema is shown as *Conceptual Specification* in Figure 5.8. Conceptually, how a monolithic organisation can be visualised using a set of interacting `OrgUnits` and how the necessary aspects of the organisation can be specified using recursive decomposition are highlighted using dotted line in the figure.

As shown in the figure, the `Organisation` is a set of `OrgUnits` where these `OrgUnits` are modular and reactive units. They have their own state, goals, behaviour and trace. They interact with each others using `Events`. The `OrgUnit` has its own `UnitState` and `UnitGoal`. The collection of `OrgUnit` specific `UnitStates` form the organisational `State`. The `Behaviours` of `OrgUnits` collectively define the organisational `Behaviour` and the aggregation of `OrgUnit` specific fragmented `Traces` form the `Trace` of an `Organisation`. An `OrgUnit` may own organisational `Measures` and an organisational `Lever` is chiefly relevant for an `OrgUnit` or a set of `OrgUnits`. The `OrgUnits` can be composed or decomposed to any level to imitate an organisation and their units (*i.e.*, organisational structure). In this formation, the `Goal` captures the *objective*, `Levers` define the *input*, `Measures` forms the *output* and other concepts describes the *model content* of a conceptual model.

For an example, the ABC University (presented in section 1.6 of Introduction chapter) can be visualised as a *University* `OrgUnit`, where the *University* `OrgUnit` is a composition of a set of *Department* `OrgUnits`. These *Departments* `OrgUnits` are typically formed using a set of *Academics* and *Students* `OrgUnits`. Each of these `OrgUnits`, *i.e.*, *University*, *Department*, *Academic* and *Student*, has its own goals and states. The `Behaviour` of a *University* `OrgUnit` is chiefly derived from the `Behaviours` of *Departments*, and the `Behaviour` of a *Department* `OrgUnit` is formed using the `Behaviours` of its constituent *Academics* and *Students*. The *Academics* are responsible for the organisational *Measures* such as number of publications, queries raised by the students and complains raised. The changes of a *University* or a *Department* can be realised by changing the formation and/or behaviour of the *Academics* and/or *Students*, *i.e.* using appropriate `Lever` definitions. This decomposition structure of ABC University is illustrated with additional details in the later part of this chapter.

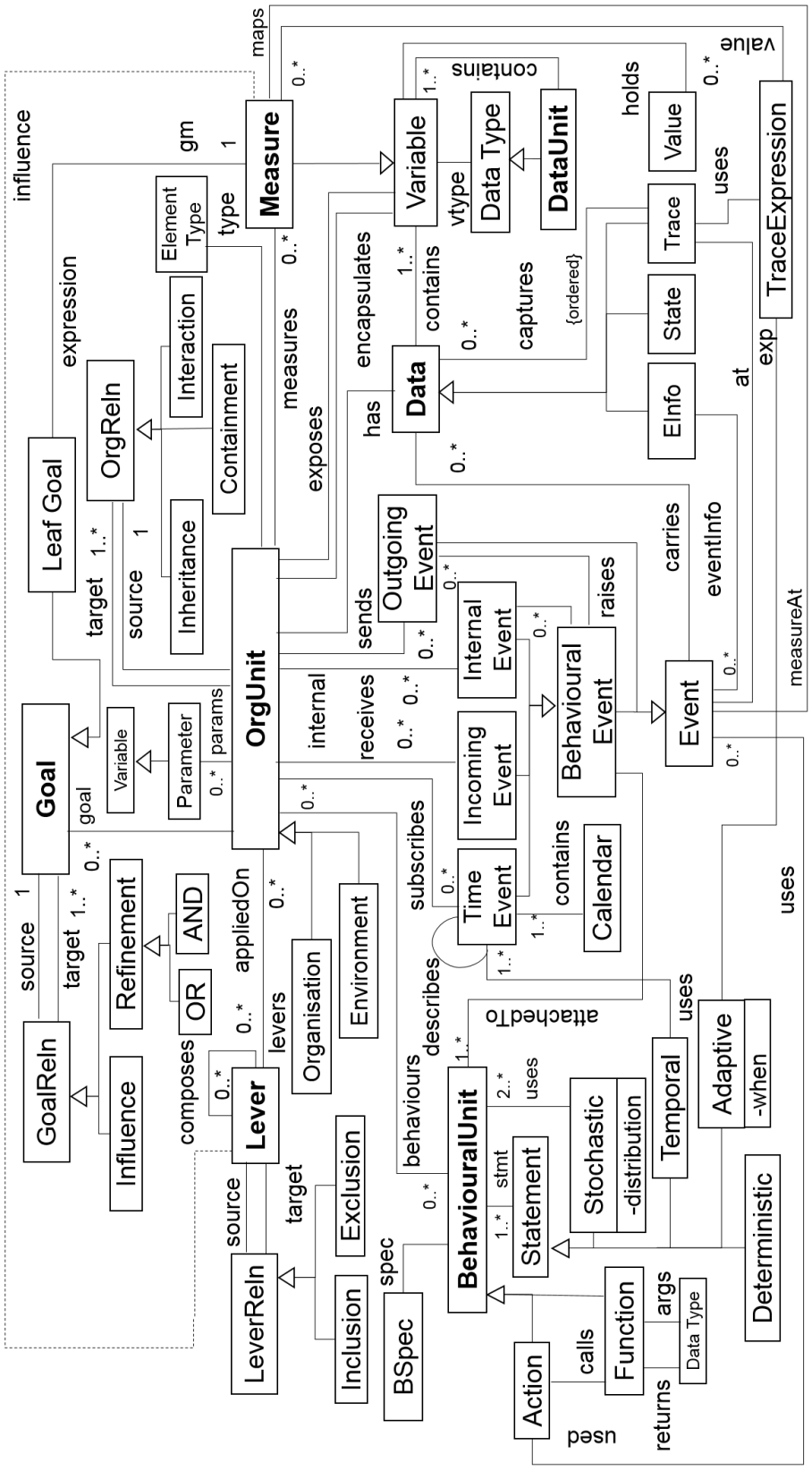


Figure 5.9 OrgML meta-model

3. **Simulation Model**: The *Simulation Model* is a simulatable specification of the information captured as *Conceptual Model*. This research considers *actor-based* language/framework as a simulation language such that the *impedance mismatch* between *Conceptual Model* (*i.e.* OrgML model) and *Simulation Model* is minimised. However, it is expected that they will differ in terms of the level of abstraction and primitive concepts. The OrgML meta-model is a Domain Specific Language (DSL) for organisational decision-making. It is capable of representing *Goals*, *Measures* and *Levers* along with the organisational concepts such as organisation, organisational units and environment in an explicit form. The actor languages and frameworks, in contrast, are general purpose language and represent complex system using the notion of actor, actor interactions or message passing, actor computation, state and trace as described in Chapter 4. ESL is used to represent the *Simulation Model* in this research, however any of the other actor languages and framework, such as Erlang [12], Scala Actor [90] and Akka [5], can be considered as a specification means for *Simulation Model*. An experiment with Akka specification [5] as simulation model is presented in Appendix D.

5.4 OrgML meta-model

OrgML meta-model refines the conceptual model presented in Figure 5.8 to capture the organisation and its environment using a set of interacting `OrgUnit`. The OrgML meta-model is depicted in Figure 5.9. As shown in the figure, `OrgUnit` is a *typed* and *parametric* element that can have: a set of typed `Variables` to characterise a `OrgUnit`, a set of `Goals` to represent its intention or objective, `Data` to capture state and trace, a set of `BehaviouralUnits` for the behaviour of the `OrgUnit`, and a set of `Events` for interactions . An `OrgUnit` typically encapsulates a set of `Variables` and it may expose a set of `Variables` to other `OrgUnit` thus relaxing data hiding or encapsulation of the pure *actor* of *actor model of computation* [2]. Typically, an `OrgUnit` *receives* a set of `IncomingEvents`, *sends* `OutgoingEvents`, *subscribes* specific set of `TimeEvents` and internally processes a set of `InternalEvents`. An `OrgUnit` can have a set of `Measures` that describe the key performance indicators. The core concepts of OrgML meta-model that specify the structural, state, behaviour, data and decision-making related aspects of `OrgUnit` are described below:

1. Structural elements

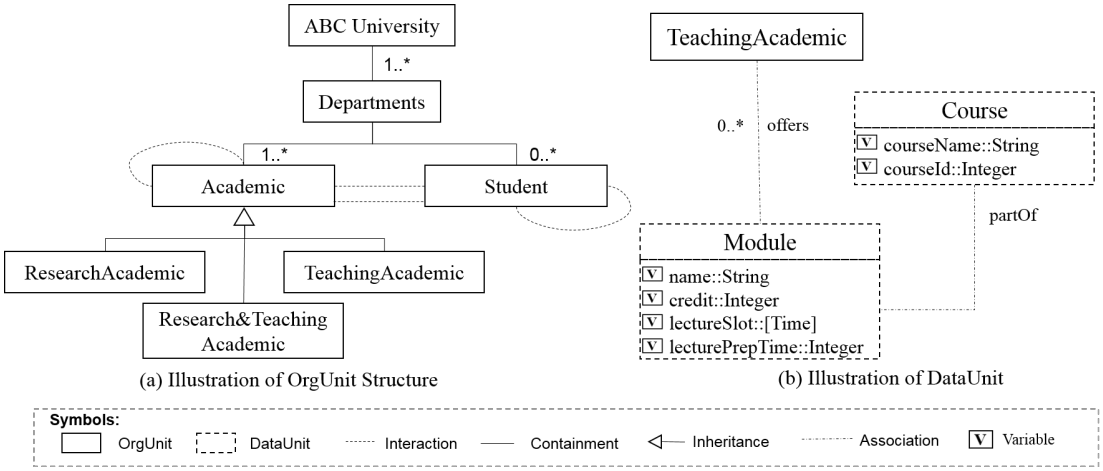


Figure 5.10 Illustration of Organisational Structure

OrgUnit Type: `ElementType` of an `OrgUnit` capture type information and describes domain semantics. For example, one can consider the *Active* and *Passive* structure as described in ArchiMate [100] as the basis for `OrgUnit` type definition. In this classification, the *Academics* and *Students* of *ABC University* can be classified as *Active OrgUnits*, whereas the *Courses*, *Modules* and *Projects* are the examples of *Passive OrgUnit* (*see* Figure 1.2 for constituent elements of *ABC University*). Alternatively, the `OrgUnit` can be categorised into three kinds based on their behaviour such as – mechanistic entity, social entity, and composite socio-technical entity. For an instance, the humans, machines and departments of an industry can be classified as social entity, mechanistic entity, and socio-technical entity respectively.

Structural Relationships: The structural topology of an `OrgUnit` is specified using `OrgReln` entity. It supports three kinds of relationships: `Containment`, `Inheritance` and `Interaction`. The `Containment` relationship expresses composition. The `Inheritance` is used to extend and override the structural and behavioural properties of an `OrgUnit`. The `Interactions` describe message passing between `OrgUnits`.

A part of *ABC University* structure is depicted in Figure 5.10 (a) as an illustration. As shown in the figure, *ABC University* is formed using a set of *Departments* wherein each *Department* contains a set of *Academics* and *Students*. The *Academics* can be categorised into three inherited `Academics`: `ResearchAcademic`, `TeachingAcademic`, and `Research&Teaching Academic`. The key interactions between *Students* and *Academics* are also shown in the figure.

DataUnit: `DataUnit` represents a collection of `Variables`. It is similar to a data structure definition of a programming language. Figure 5.10 (b) shows two illustrative `DataUnit` –

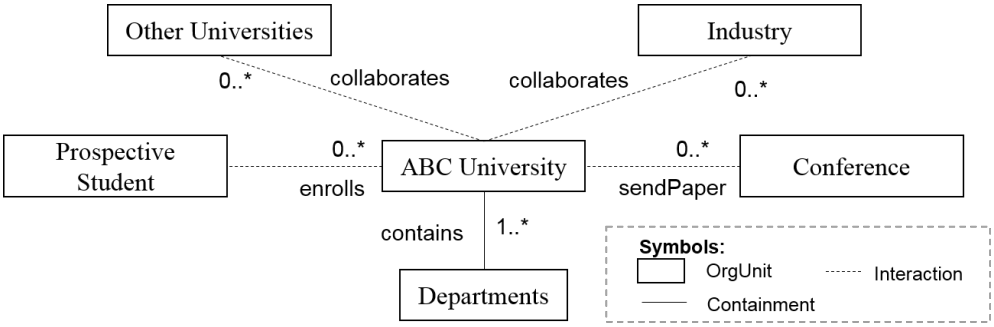


Figure 5.11 Illustration of Organisation and Environment

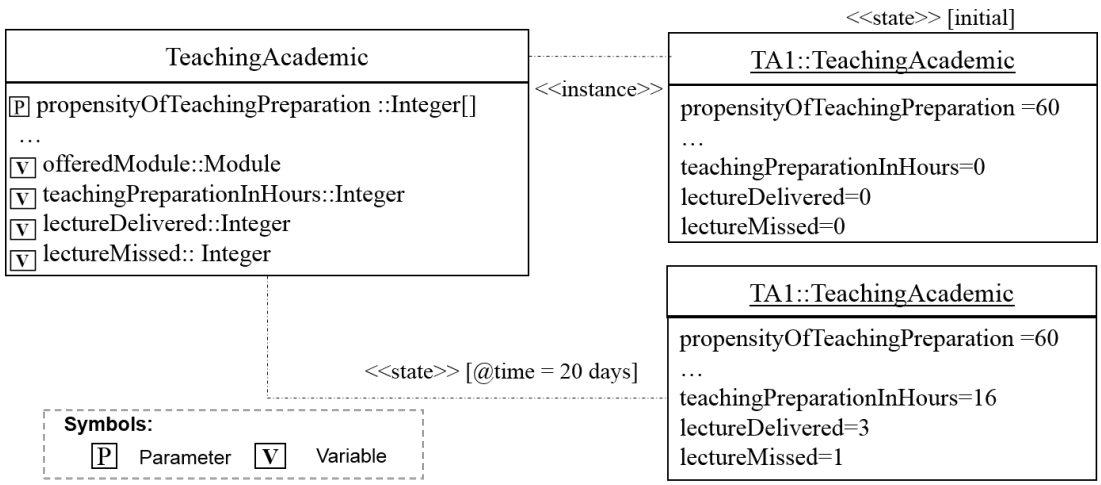


Figure 5.12 Illustration of OrgUnit Variables, Parameters and State

Module and *Course*. The *Course* `DataUnit` describes course details using two typed `Variables`: *courseName* and *courseId*, and *Module* `DataUnit` describes the course modules using four typed `Variables` – *moduleName*, *credit*, *lectureSlot* and *lecturePreparationTime*.

Organisation and Environment: Both, the `Organisation` and `Environment` cannot be composed within an `OrgUnit`. However, they can interact with other `OrgUnit` and decompose into finer level of granularity. For example, an `Organisation` can interact with the `Environment` and `Organisation` can be decomposed into organisational units. In the context of *ABC University*, the *ABC University* is an `Organisation` and other universities, prospective students, conference venues, industries are the `Environment` of *ABC University*. A conceptual schema describing the interactions of *ABC University* with its `Environment` and its containment relationships are shown in Figure 5.11.

Variable and Parameter: `Variable` and `Parameters` are *typed* entities that represent the property or state variable of an `OrgUnit`. OrgML meta-model supports *Integer*, *String*, *Double*,

Date primitive *data types*, and considers `OrgUnit` and `DataUnit` definitions as composite types. The `Variables` with `OrgUnit` data types form the `Containment` relationship of an `OrgUnit`. The `Parameters` need to be set to appropriate `Values` while instantiating an `OrgUnit`. to characterise an `OrgUnit`.

Figure 5.12 shows examples of `Variables` and `Parameters` of *TeachingAcademic* `OrgUnit`. As shown in the figure, '*propensityOfTeachingPreparation*' is a `Parameter` of *TeachingAcademic*. It indicates the probability for preparing a lecture at a given time. The `Variables`: '*teachingPreparationInHours*', '*lectureDelivered*', and '*lectureMissed*' describe the state of a *TeachingAcademic*. In particular, '*teachingPreparationInHours*' indicates hours spend on preparing lectures, '*lectureDelivered*' indicates number of lectures delivered, and '*lectureMissed*' describes the number of lectures missed by a *TeachingAcademic*.

2. State information

State: `State` of an `OrgUnit` is formed based on the `Values` of the state `Variables`. The initial *State* and *State* at 20 day (of an academic year) of `TA1` *TeachingAcademic* is illustrated in Figure 5.12. The figures shows, a *TeachingAcademic* named *TA1*, who has 60% probability of spending time on lecture preparation, have spend 16 hours in lecture preparation, delivered 6 lectures and missed one lecture after 20 days from the starting of an academic year.

3. Behavioural elements

Event: `OrgUnits` interact with each other through `Events`. The `Events` are classified into two categories: `OutgoingEvent` and `BehaviouralEvent`. An `OrgUnit` sends `Data` to other `OrgUnit` through `OutgoingEvents`. In constrast, `OrgUnit` performs specific behaviour when it receives a `BehaviouralEvent`. The `BehaviouralEvents` has its own behaviour, which is specified using `BehaviouralUnit`. The `BehaviouralEvent` is further classified into three kinds of events: `IncomingEvent`, `InternalEvent` and `TimeEvent`. The `IncomingEvents` are the `Events` that are received from other `OrgUnits`, `InternalEvents` are triggered internally, *i.e.*, from a `BehaviouralEvent` of own `OrgUnit`. The `TimeEvents` are the global time events. The `OrgUnits` receive subscribed `TimeEvents` and perform specific behaviour when they receive a `TimeEvent`.

A set of `Event` specification and interactions between *TeachingAcademic*, *ResearchAcademic* and *Students* are shown in Figure 5.13. As shown in the figure, the *TeachingAcademic* has an `InternalEvent` named as *TeachingPreparation*, raises *EvaluateStudent* as an

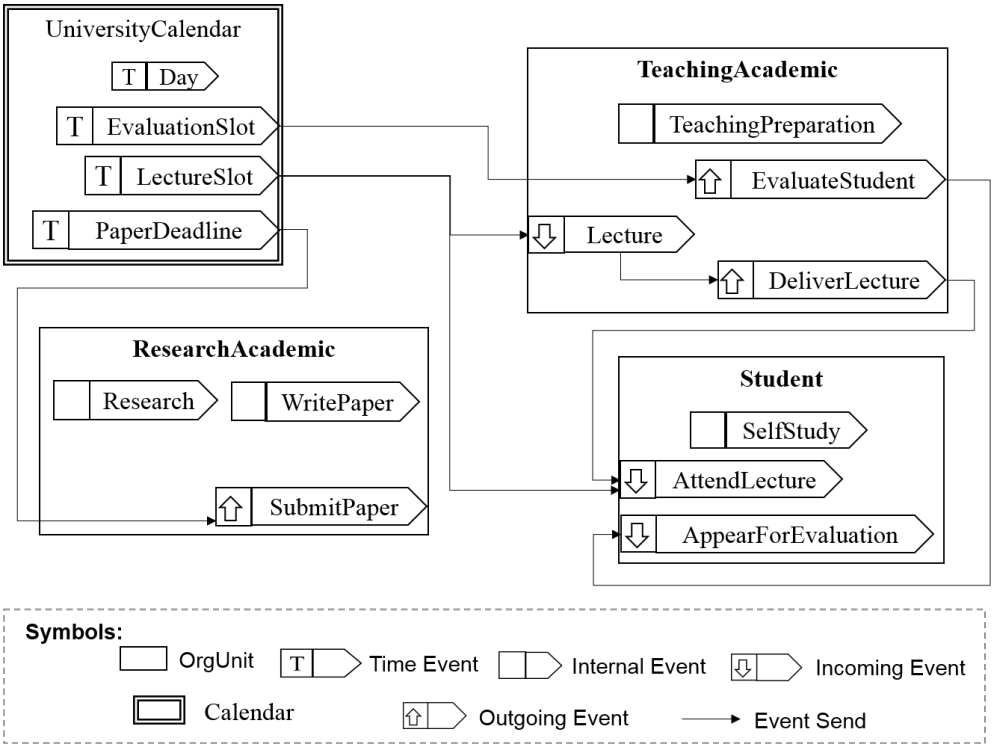


Figure 5.13 Illustration of Events

`OutgoingEvent` based on the *EvaluateSlot* `TimeEvent`, and raises *DeliverLecture* `OutgoingEvent` while *Lecture* `IncomingEvent`. Similarly, a *ResearchAcademic* internally initiates *Research* and *WritePaper* `InternalEvent` and may decide to submit paper based on *PaperDeadline* `TimeEvent`. A *Student* typically focuses on *SelfStudy* as an `InternalEvent`, *AttendLecture* based on *LectureSlot* `TimeEvent` and the *DeliverLecture* `OutgoingEvent` of *TeachingAcademic* `OrgUnit`.

Behaviour: The behaviour of an `OrgUnit` is described using `BehavouralUnits`. They are a sequence of `Statements` where each `Statement` can create *new* `OrgUnit`, raise `InternalEvent`, raise `OutgoingEvent`, update `Variables` and perform other computations. These `Statements` can be categorised into four kinds – `Deterministic`, `Stochastic`, `Temporal` and `Adaptive`. Standard language constructs, such as *assignment*, *condition*, *loop*, and *message passing*, are `Deterministic Statements`. The `StochasticStatement` specifies the uncertainty in performing specific `Statements`. The `TemporalBehaviour` uses `TimeEvent` to express temporal relationships within behavioral specification. `AdaptiveBehaviour` describes adaptation rules. It expresses the behavior which is activated when a specific condition is matched – it uses `TraceExpression`, *i.e.*, an expression over `Trace` element, to define the conditions. The

```
0  /* Legends: OrgML Keywords,   OrgML Meta Elements   */
1  action ::= on event  where state do { stmt* }        Action specification
2
3  function ::= { stmt* }                               Function specification
4
5  event ::= p_event                                    Primitive Event
6        | time                                         Time Event
7        | event[exp]                                   Number Of Occurrence
8        | no event                                     Event Not Occurred
9        | { event* }                                   Any Event From List
10       | event between [event,event]                  Event Between Two Events
11       | [event*]                                     Sequence Of Events
12       | [[event*]]                                   Strict Sequence
13
14 p_event ::= id(type*)                                Event definitions
15
16 state ::= {exp* }                                    List Of Conditions
17
18 exp  ::= lvar                                        Variable
19       |  integer | boolean | string | float | date   Constants
20       |  null                                        Undefined
21       |  exp op exp                                  Binary expression
22       |  not exp                                    Negation
23       |  fun(exp*)                                  Function Call
24       |  [ exp* ]                                   List Of Expressions
25       |  []                                         Empty Expression
26
27 stmt  ::= decl                                        Local Variables
28       |  lvar := exp                                 Assignment
29       |  new id(exp*)                                Create New OrgUnit
30       |  for (lvar:exp) do stmt                       Looping
31       |  { stmt* }                                   Block Statement
32       |  if exp then stmt else stmt                  Conditional Statement
33       |  p_event(exp) → id                           Send Event
34       |  probably(exp) stmt else stmt                Uncertainty
35
36 type ::= id                                           Type OrgUnit, DataUnit
37       |  Integer | Boolean | String | Float | Date    Primitive Types
38       |  Void                                        Undefined
39       |  [ type ]                                    List Type
40
41 lvar ::= variable                                    OrgUnit Variable
42     | decl                                           Local Variable
43
44 decl ::= id :: type                                   Declare Local Variable
45
46 time ::= p_time                                       Primitive Time
47     | ( time)                                        Grouping
48     | time( integer ) of time                        Every nth Occurrence
49     | time except time                               Not Of TimeEvent
50     | [ time* ]                                      Sequence Of TimeEvent
51     | anytime [time* ]                               Anytime from a list Of TimeEvent.
```

Figure 5.14 Syntax of BSpec specification

```
no (E)            ⊨ ¬(E)                              Negation [Line 8 in BSpec Syntax]
{E1, E2, ... , En} ⊨ (E1 ∨ E2 ∨ ... ∨ En)             Alternate  [Line 9]
E between[E1,E2]  ⊨ (E1 ∧ ◊ (E ∧ ◊ E2))               Between [Line 10]
[E1,E2, ... En]   ⊨ (E1 ∧ ◊ (E2 ∧ ◊ ( ... ∧ ◊ (En)))) Sequence [Line 11]
[[E1,E2, ... En]] ⊨ (E1 ∧ ○ (E2 ∧ ○ ( ... ∧ ○ (En)))) Strict Sequence  [Line 12]
```

Figure 5.15 Semantics of event specification

element `BSpec` is a placeholder for textual behavioural specification. A high-level syntax of proposed textual behavioural specification is shown in Figure 5.14. The syntax of the `Statement` specification is described using *stmt* rule (line 27 – 34 of Figure 5.14). It supports local variable declaration (line 27), assignment statement (line 28), 'new' operator (line 29), for loop (line 30), block statement (line 31), conditional statement (line 32), message passing (line 33) and probabilistic statement (line 34).

```
0  /* Legends:  OrgML Keywords ,    OrgML Meta Elements    */
1  ActOnQueries::Action = on StudentQuery[4] do {
2    //Resolve queries by raising QueryResolution Event to all four students.
3  }
4
5  ActOnLectureSlot::Action = on (no {Complain, StudentQuery[4]}) between [Day, LectureSlot] do
       {
6    //Lecture a module by raising DeliverLecture Event
7  }
8
9  Calendar {    //Example of TimeEvent Specification
10  Hour= primitive          //Primitive Event
11  Day= Hour(8)             //Raise a Day TimeEvent on every 8 Hour TimeEvents
12  Week= Day(5)             //Raise a Week TimeEvent on every 5 Day TimeEvents
13  Month= Day(22)           //Raise a Month TimeEvent on every 30 Day TimeEvents
14 }
15 //  A complex expression that indicate  raise two LectureSlots one on every third hour of
       second working day of a week, and second one on every fifth hour of forth working day
       of a week expect first working day of a month
16 LectureSlot = [ (Hour(3) of Day(2) of Week), (Hour(5) of Day(4) of Week except Day(1) of
       Month)]
17
18
19 //Alternate Definition of ActOnLectureSlot (A Lever specification)
20 ActOnLectureSlot::Action = LectureSlot where (teachingPreparationInHours > 0) do {
21   //Lecture a module by raising DeliverLecture Event
22 }
```

Figure 5.16 Example of Action and TimeEvent specifications

As shown in Figure 5.9, the `BehavouralUnit` is specialised into two types of unit – `Function` and `Action`. A `Function` is a behaviour unit that contains a coherent set of `Statements` (syntax is shown in line 3 of Figure 5.14). These `Functions` must be called from `Statements` as shown in line 23. The `Action` is a behaviour unit that triggers when a complex event specification is satisfied. The syntax of a supported complex event specification is shown as *action* rule in Figure 5.14. The specification supports *repetition* (line 7 of Figure 5.14), *negation* (line 8), *alternative* (line 9), *between* (line 10), *sequence* (line 11) and *strict sequence* (line 12). The semantics of the supported constructs are established using Linear Temporal Logic (LTL) [158] as shown in Figure 5.15. For example, the *negation* of an event `E` is implied to be true if event `E` is not occurred till now. The *alternative* of a set of events (*i.e.*, { E1, E2,...,En} implies at least one event is occurred till now. Similarly, the semantics of *between*, *sequence* and *strict sequence* are defined using LTL formulae. The formulae are evaluated in simulation runs using pattern matching over event traces.

Figure 5.16 highlights complex event specifications of two `Actions` of *TeachingAcademic*. The `Actions` are: '*ActOnQueries*' and '*ActOnLectureSlot*'. The definition '*ActOnQueries*' (shown in line 1 –3) describes that a *TeachingAcademic* accumulates four '*StudentQuery*' `IncomingEvents` to collectively act on them. Whereas the definition of '*ActOnLectureSlot*' (line 5–7) states that a *TeachingAcademic* delivers a lecture (*i.e.*, raises '*DeliverLecture*') when following conditions are satisfied – (i) a '*LectureSlot*' is raised, (ii) no '*Complaint*' is raised on the same *Day*, and (iii) the number of '*StudentQuery*' raised on the same *Day* is less than four.

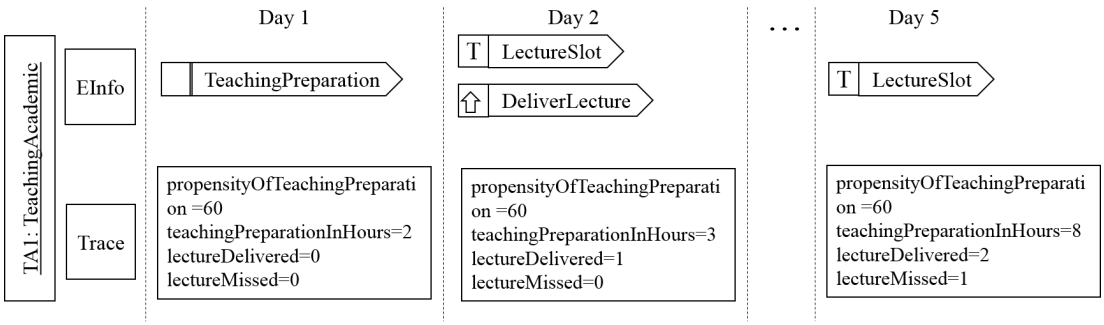


Figure 5.17 Illustration of Data and Traces

Calendar: `Calendar` is an entity that contains global `TimeEvents` wherein each `TimeEvent` indicates significant time, such as *Day*, *Month*, *Beginning of Academic Year*, *End of Academic Year*. An illustration of `Calendar` entity: *UniversityCalendar* is shown in Figure 5.13. As shown in the figure, *UniversityCalendar* contains four `TimeEvents` – '*Day*', '*EvaluationSlot*', '*LectureSlot*', and '*PaperDeadline*'. The '*Day*' event indicates the beginning of a day, '*EvaluationSlot*' indicates the time slots for module evaluations, '*LectureSlot*' indicates a set of time slots for lecturing a module, and '*PaperDeadline*' indicates the various paper deadlines. These `TimeEvents` could be either be a *primitive* time event (raised by underlying simulation engine) or a derived time event, which is an expression over other time events. A syntax to describe derived time events is highlighted in line 46–52 of Figure 5.14. The examples of derived time expressions are shown in line 9–16 of Figure 5.16 where the specification states that a *Day* is 8 *Hours*, a *Week* is 5 *Days*, and a *Month* is 22 *Days*. Specification also highlights a complex time expression that schedules a lecture slot twice in a week – (i) third hour of second day of every week (*i.e.*, every Tuesday 11 AM considering the working hour starts at 8 AM), and (ii) fifth hour of forth day of a week excluding the day if forth day of a week is the first day of a month (*i.e.*, every Thursday 3 PM excluding the day if month starts with Thursday).

4. Data elements

Data and Trace: `Data` of an `OrgUnit` is a set of typed `Variables` that capture three elements: `State`, `EInfo`, and `Trace`. `EInfo` captures the `Events` produced internally, `Events` communicated to other `OrgUnits`, and `Events` received by an `OrgUnit` along with the time information. The `Trace` is a sequence of `Data` from any time in the past. As an illustration, a set of `States`, `EInfo` and `Trace` of a `TeachingAcademic` are highlighted in Figure 5.17. The figure shows a sequence `Data` of *TA1* `TeachingAcademic`, which are captured on *Day* `TimeEvents`. At *Day*

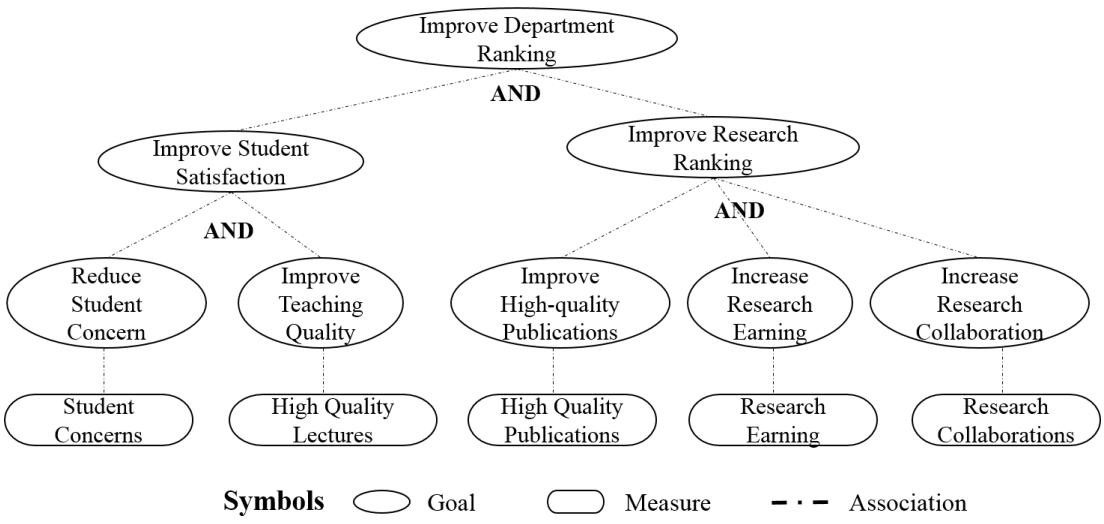


Figure 5.18 Illustration of Goal, Goal structure and Goal-to-Measure relationship

1, the *TA1* `TeachingAcademic` triggers *TeachingPreparation* and spend 2 hours on preparation activity. Subsequently, *TA1* `TeachingAcademic` delivers a lecture on *Day 2* and misses a lecture on *Day 5*.

5. Decision making concepts

Measure: `Measures` are specialised `Variables` that describe the key performance indicators of an `OrgUnit`. The `Measures` of a `TeachingAcademic` are – number of lectures taken in an academic year, number of lecture missed due to some high priority work, *etc*. The `Measures` are connected to `Trace` using `TraceExpression`. The `Measures` are linked to `Traces`, `Traces` are sequence of `Data` and `Data` hold `Values`. Therefore, the value of a `Measure` can be computed by navigating `Measures`, `Trace`, `Data` relationships.

Goal: The `Goal` of an `OrgUnit` captures its intention or a set of objectives. A `Goal` can be hierarchically decomposed into sub-`Goals` and a `Goal` can influence other `Goals`. These relationships can be specified using `GoalReln` wherein the `Refinement` relationship captures goal decomposition relationship and the `Influence` relationship specifies the goal influence relationship. The `Refinement` can be further classified into two kinds of decomposition: `And` decomposition and `Or` decomposition. In a goal hierarchical structure, the leaf level goals (termed `LeafGoals`) are linked to appropriate `Measures` so that they can be quantitatively measured. In particular, a `LeafGoal` is a conditional expression over `Measures`. Therefore, the

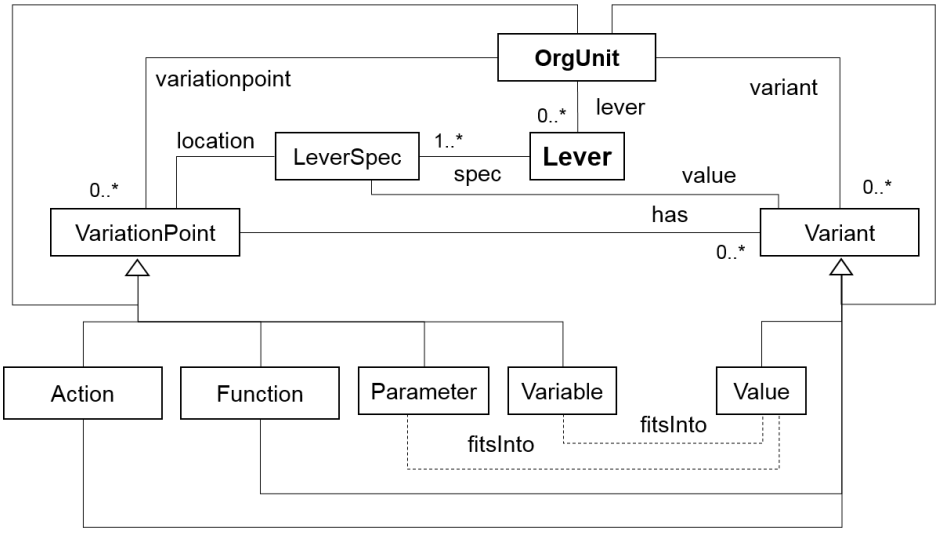


Figure 5.19 Lever definition specification

`LeafGoal` can be computed using associated `Measure` values and other `Goals` can be computed using bottom-up goal navigation and evaluation.

An example of `Goal`, `Goal` decomposition structure, and mapping between `LeafGoals` and `Measures` are shown in Figure 5.18. As shown in the figure, the *Research&TeachingAcademic* has a primary `Goal` to '*Improve Departmental Ranking*' wherein the goal '*Improve Departmental Ranking*' can be achieved by achieving '*Improve Student Satisfaction*' *And* '*Improve Research Ranking*'. The goal '*Improve Student Satisfaction*' is further decomposed of two `LeafGoals`: '*Reduce Student Concerns*' and '*Increase High Quality Lecture*'. The goal '*Improve Research Ranking*' is a decomposition of three `LeafGoals`: '*Improve High Quality Publications*', '*Increase Research Earnings*' and '*Increase Research Collaboration*'.

Lever: `Lever` represents possible courses of action that can be applied on an `OrgUnit`. A `Lever` can be decomposed. A lever specification contains two kinds of specification: (i) lever usage specification and (ii) lever definition. `Lever` usage specification is illustrated in OrgML meta-model (depicted in Figure 5.9) using `LeverReln` and its specialisation. The `Lever` inclusion and exclusion relationships can be defined using `LeverReln`.

`Lever` specifies changes in either `Data`, `Structure`, behaviour of an `OrgUnit` or a combination thereof. This research adopts the notion of *variability modelling* [122] and uses the concept of `Variation Point` and `Variant` to define lever specification. The concept of variability modelling with respect to the OrgML meta-model is depicted in Figure 5.19. As shown in the figure, each `OrgUnit` may have multiple `VariationPoints` and each `VariationPoint` of an

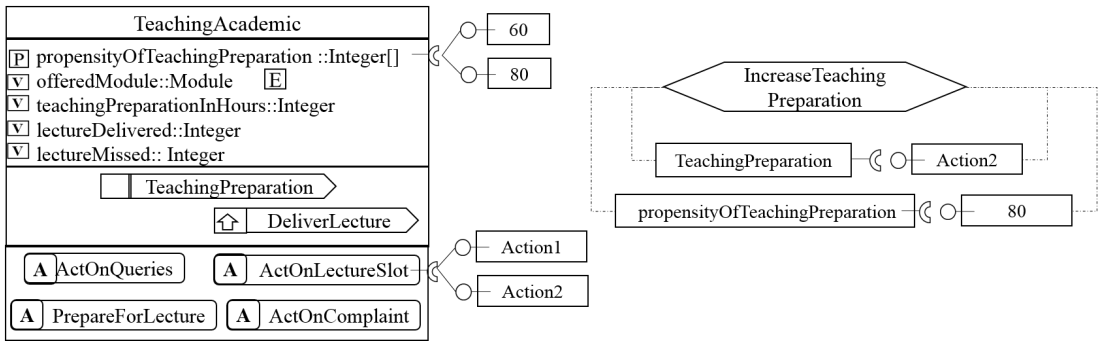


Figure 5.20 Example of Lever specifications

`OrgUnit` should have a set of alternative `Variants`. A `VariationPoint` is an element of an `OrgUnit` which is amenable for a change, and a `Variant` describes the change specification that can be fitted into a `VariationPoint`. A predefined set of core elements of OrgML can act as `VariationPoint` and `Variant`. Further, there is a notion of compatibility between `VariationPoint` and `Variant`. As shown in the figure, `Parameter`, `Variable`, `Action`, `Function` and `OrgUnit` of an OrgML model can act as `VariationPoints` wherein the element `Value` can fit into `Parameter` and `Variable`; element `Function` can fit into `Function`; element `Action` can fit into `Action`; and an `OrgUnit` can fit into an `OrgUnit`.

In this structural formation, a `Lever` is set of `LeverSpec`, where each `LeverSpec` selects `Variant` for one or more `VariationPoint`(s). Figure 5.20 highlights the concept of `VariationPoint`, `Variation` and `Levers`. As shown in the figure, the `Parameter` '*propensityOfTeachingPreparation*' is a `VariationPoint` with two `Value` `Variants` (*i.e.*, 60% and 80% propensity for teaching preparation). Similarly, the `Action` '*ActOnLectureSlot*' is a `VariationPoint` with two `Action` `Variants`: '*Action1*' and '*Action2*'. '*Action1*' `Variant` considers addressing students complaint and student queries as high priority activities than delivering a lecture as specified in line 1–3 of Figure 5.16 whereas '*Action2*' `Variant` considers delivering lecture as a high priority activity than addressing complaint and queries (as specified in line 20–22 of Figure 5.16). The figure shows a `Lever` specification named as '*Increase Teaching Preparation*'. `Lever` selects `Value` 80 (%) for '*propensityOfTeachingPreparation*' and *Action2* as '*TeachingPreparation*' action.

Discussion

The concepts introduced in the proposed meta-model is grounded in well understood theories in the research literature and established practice. For example, the decomposition, moduarisation

Table 5.1 Conceptual mapping with existing specifications

OrgML Concept	**Concepts from existing specification languages**
OrgUnit	**UML Class Diagram** :: Class that represents Organisational elements such Organisation, Organisational Unit, Environment. **ArchiMate**:: Business Actor, Business Role, Business Object, Application Component, System Software
Data	**UML Class Diagram**:: Class that represents entities **ArchiMate**:: Data Object, Artifacts. **BPMN**:: Data Object
Goal	**i* specification**:: Goal. **ArchiMate**:: Meaning.
Behaviour	**UML State Machine**:: State, Transition **ArchiMate**:: Business Service, Business Process, Business Function, Application Function, Infrastructure Function. **BPMN**:: process definition.
Event	**UML State Machine**: Transition. **BPMN**:: Event. **ArchiMate**:: Business Interaction, Business Event
Measure	**i* specification**:: Task, Leaf level Goal. **BPMN**:: KPI

and unit hierarchy of *OrgUnit* are taken from the notion of the component abstraction [32]. The goal-directed reactive and autonomous behaviour are traced to *actor* behaviour [96]. An event driven architecture [139] is adopted to introduce reactive behaviour. The concept of intentional modelling is adopted to enable specification of goals [218]. The behavioural classification and uncertainty is derived from the notion of *known* and *known unknown* uncertainty classification coined by Donald Rumsfeld [170]. The concepts introduced in this meta-model also relate to a range of Enterprise Modelling (EM) specifications. The conceptual mapping of the OrgML concepts with the concepts defined in EM related literature are illustrated in Table 5.1.

The proposed OrgML meta-model realises the *Constructs* derived from management literature (depicted in Figure 3.6) and satisfies the modelling and analysis requirements illustrated in Table 3.3 (of Chapter 3). OrgML meta-model explicitly supports the decision-making concepts, such as `Goal`, `Measure` and `Lever`. Likewise, the `Event` definition, `Data`, and `OrgUnit` structure collectively specify the *what* aspect, `OrgUnit` help specify the *who* and *where* aspects, `Goal` specification specifies the *why* aspect, and the `BehaviouralUnit` along with the `Event` specification specify the *how* and *when* aspects.

The concept `OrgUnit` enables the modelling of complex organisation using a set of hierarchically composable `OrgUnits` each listening/responding/raising events of interest. Each `OrgUnit` encapsulates state (*i.e.*, a set of State variables), trace (*i.e.*, data along with the events that it has responded to and raised till now) and behaviour (*i.e.*, encoding of individual reactions). Therefore, the concept `OrgUnit` ensures the required modularity and encapsulation. An `OrgUnit`

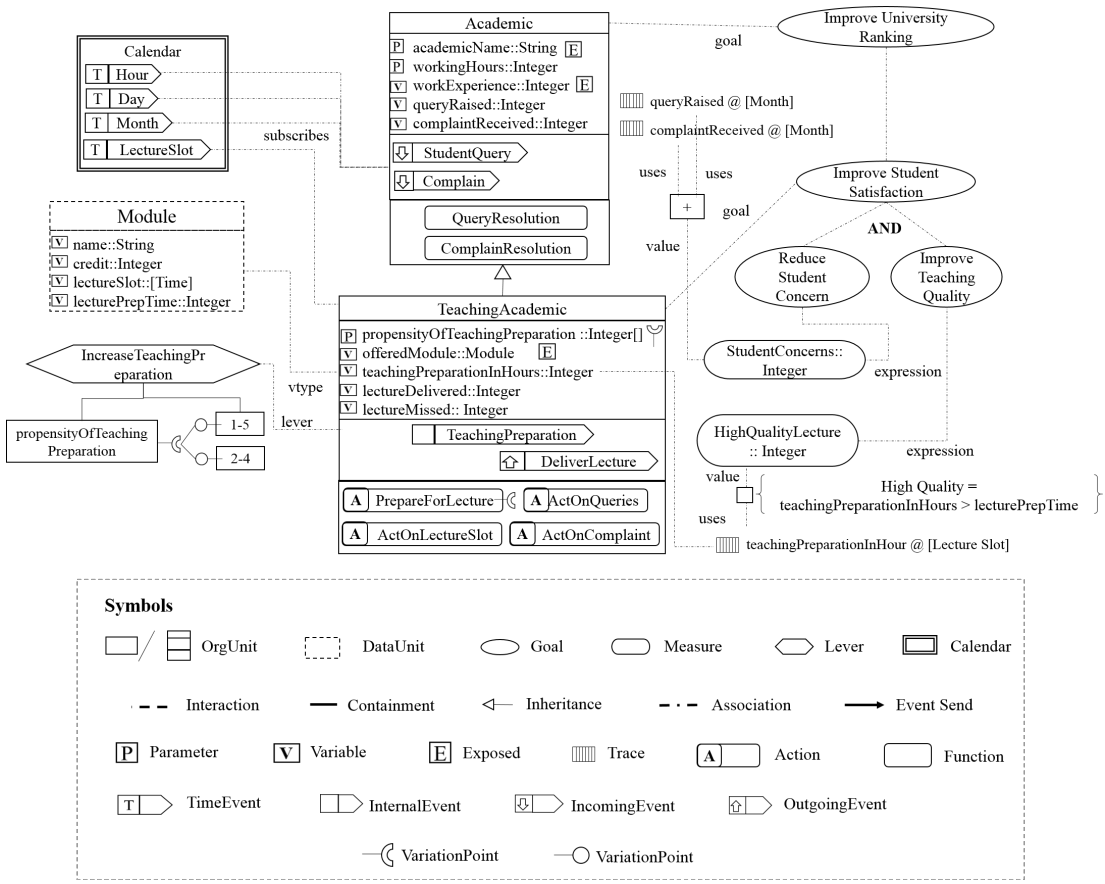


Figure 5.21 An OrgML model

interacts with each other `OrgUnits` through a set of Events. The interactions between `OrgUnit` helps to specify reactive behaviour, which gives rise to the emergent behaviour in a overall actor topology.The `InternalEvent` and `TimeEvent` collectively specify the autonomous behaviour, *Stochastic* behaviour provides uncertainty, the *Temporal* behaviour and `TimeEvent` specify the temporal behaviour. Altogether, the proposed OrgML supports a top-down approach for defining organisational goals, a middle-out approach for defining structural aspect of an organisation, and a bottom-up approach for behavioural specification.

Therefore, this research argues that the proposed OrgML meta-model is a domain specific language for organisational decision-making and also grounded with prevalent enterprise modelling related concepts. The concepts introduced in the OrgML meta-model are represented using a set of notations defined in Appendix B. The utility, efficacy and expressiveness of the proposed OrgML meta-model are evaluated using case studies in Chapter 7.

Table 5.2 OrgML to ESL transformation strategy

OrgML Concept	ESL Concept	Sample Translated ESL Spec.
OrgUnit, Organisation, Environment	Actor	act teachingacademic(…)::TeachingAcademic {…}
DataUnit	Actor	act module()::Module {…}
Parameter	Actor Variables and input parameters in 'new' operator	act teachingacademic(p_academicName::Str, p_workingHour::Int,…) :: TeachingAcademic { academicName::Str = p_academicName; …}
Encapsulated Variable	Actor Variable	1. workExperinece::Int= 0; 2. queryRaised::Int = 0;
Exposed Variable	Exported Actor Variable	export adacemicName,…;
Trace	Actor Variable with List data-type	trace_queryRaised::[Int] = [];
IncomingEvent	Event	1. StudentQuery(…) → {...}; 2. Complain(…) → {...};
OutgoingEvent	***send*** Event	calender ← RegieterForLectureSlot(self);

TimeEvent	Event	1. Day(day::Int) → {...}; // same as IncomingEvent
		2. Month(month::Int) → {...};

InternalEvent	Event	TeachingPreparation(…) →{...};
Function	Function	evaluate_teachingQuality(…)::Bool = if (teachingPreparationInHours > lecturePrepTime) then true else false;
Action	ESL specification using Actor, Event and Function	act actPrepareForLecture()::ActPrepareForLecture {...} PrepareForLecture(…) → {...};
Calendar	Actor with Event specification	act calendar()::Calendar { hour::Int = 0; hourSubscriber::[T] = []; ... Hour(hour::Int) → {};}

Inherited OrgUnit	Actor (it considers OrgML inheritance hierarchy)	act teachingacademic(…)::TeachingAcademic {
		// Translated specification of all OrgUnits in inheritance hierarchy.
		//translation of own specification}

Uncertainty	probably (probability) ← {statement}	probably (probability) ← {statement}
Goal	Not supported	-
Measure	ESL specification using Function and Display	Dashboard ← Display(evaluate_teachingQuality(…))
Lever	Event specification and ESL Statements	LeverIncreaseTeachingPreparation → { propensityOfTeachingPreparation = [1,5] }
Statement	ESL Statement	queryRaised := queryRaised+1;

5.5 Transformation of OrgML to simulation language

This research has proposed the use ESL as a language for simulation specification to perform quantitative *what-if* analysis. This section presents an one-way model transformation strategy to translate OrgML model that conforms to OrgML meta-model (as shown in Figure 5.9) to ESL constructs. The proposed translation strategy is illustrated using a subset of University case study. The key transformation rules are described using a Xtend model transformation template language [39] in Appendix C.

Although ESL has been used as the simulation language, this research claims that other actor languages and frameworks, such as Erlang [12], Scala Actor [90] and Akka [5], can also be considered as simulation specification. As a justification to this claim, a transformation strategy to transform OrgML specification into Akka specification [5], which is a Java based industry-scale actor framework, is presented in Appendix D.

A subset of University case study

An OrgML specification describing a subset of University case study is presented in Figure 5.21 to discuss the OrgML to ESL transformation strategy and translation rules. The model shown in the figure is an integrated view of a series of fragmented models presented in Figure 5.10 to Figure 5.20 in section 5.4. The model contains *TeachingAcademic* `OrgUnit` that inherits from *Academic* `OrgUnit`, a *Module* `DataUnit`, a `Calendar` with four `TimeEvents`. The `Parameters`, `Variables`, `IncomingEvents`, `OutgoingEvents`, `InternalEvents`, `Traces`, `Goals`, `Measures` an `Lever` of *Academic* and *TeachingAcademic* are shown as discussed in section 5.4.

An overview of OrgML to ESL transformation strategy

A high-level transformation strategy that describes the concept mapping along with sample ESL code fragments are presented in Table 5.2. Conceptually, the `OrgUnit` and its specialisation, *i.e.*, `Organisation` and `Environment`, are mapped onto ESL *Actor*. The `DataUnit` is realised as ESL *Actor* without any behaviour. `Calendar` is realised as ESL *Actor* with `TimeEvent` related behaviours. The interactions among `OrgUnits` are mapped onto event specifications.

The constituent elements of `OrgUnit`, `DataUnit` and `Calendar` are translated into the elements of ESL Actor. The OrgUnit `Parameters` are translated into ESL Actor variables; all exposed and encapsulated `Variables` are translated in ESL actor variables; and `Traces` are translated into ESL actor variables with list data-type. The `IncomingEvents`, `InternalEvents` and `TimeEvents` are translated into ESL event specification. All OrgML `Statements` that describe the behavioural specification of OrgML `Functions`, `Actions` and `Behavioural Events` are translated into ESL specification by translation OrgML `Statements` into ESL specification. The OrgML `Statements` are the specifications that conform to the behavioural *stmt* syntax presented in line 27 – 34 and *exp* rules presented in line 18–25 of Figure 5.14. These statements specify variable assignments, *new* actor, looping, conditional statement, *send* event,

and *probably*, which are also supported in ESL. Therefore, all statements can be mapped to ESL statements by suitable syntactic transformation.

The `Measures` are mapped onto the ESL *Variables* and `Levers` are converted into ESL Event specifications. Principally, the `Levers` are the change specification of OrgML `Data`, `Event`, `Function`, `Action` and `OrgUnit` and their combination as shown in Figure 5.19. Therefore, an OrgML `Lever` definition can be translated into ESL specification by applying multiple OrgML concept to ESL transformation rules in specific sequence. Detailed transformation rules are presented in Appendix C.2.

OrgUnit inheritance is resolved by translating inherited `OrgUnits` into ESL actors such that each inherited `OrgUnit` includes its own and inherited `Variables`, `Parameters`, `Events`, `Functions` and `Actions`. Conceptually, it adopts '*one data entity for concrete class*' pattern defined for object to relational tables mapping [107]. The transformation logic considers the following overriding and overloading rules:

- `Variables` and `Parameters` cannot be overridden in an inherited `OrgUnit`.
- `Events` and `Functions` can be overridden and overloaded.
- `Actions` can be overridden but cannot be overloaded.

The precise rules to convert an inherited `OrgUnit` to an ESL actor is described in Appendix C.2.3.

As an illustration of the transformation strategy, the key elements of the translated ESL specification of the OrgML model depicted in Figure 5.21 are shown in Figure 5.22. As shown in the figure, the inherited *TeachingAcademic* `OrgUnit` is translated into *teachingacademic* ESL actor specification. Actor *teachingacademic* considers all `Parameters` of *TeachingAcademic* and *Academic* as actor parameters (line 1), exports all exposed `Variables` of *TeachingAcademic* and *Academic* (line 3), and contains all `Variables` of *TeachingAcademic* and *Academic* as actor variables (line 12–14, 16, 19, *etc.*). It subscribes `TimeEvents` by sending registration request to '*calendar*' actor as shown in line 38–39 and it realises OrgML `Lever` specification using ESL event definition as shown in line 65.

In addition, the translated *teachingacademic* actor contains the following elements:

- Trace variables to capture traces of *TeachingAcademic* and *Academic* `OrgUnits` (line 17, 20, *etc.*).

```
0  /* Legend: ESL Keywords */
1  act teachingacademic(p_academicName::Str,p_workingHour::Int,p_propensityOfTeachingPreparation
       ::Int)::TeachingAcademic {
2
3   export adacemicName,workExperinece,offeredModule;
4
5   queryResolution()::Bool = { ... };
6   complaintResolution()::Bool = { ... };
7
8   evaluate_studentConcerns()::Int = nth(trace_queryRaised,length(trace_queryRaised))  + nth(
       trace_complaintRecieved,length(trace_complaintRecieved));
9
10  evaluate_teachingQuality()::Bool = if (teachingPreparationInHours > offeredModule.
       lecturePrepTime) then true else false;
11
12  academicName::Str = p_academicName;
13  workingHour::Int= p_workingHour;
14  propensityOfTeachingPreparation::Int = p_propensityOfTeachingPreparation;
15
16  workExperinece::Int= 0;
17  trace_workExperinece::[Int]= [];
18
19  queryRaised::Int = 0;
20  trace_queryRaised::[Int] = [];
21  ...
22  offeredModule::Module = [];
23  trace_offeredModule::[Module] = [];
24
25  teachingPreparationInHours::Int = 0;
26  trace_teachingPreparationInHours::[Int] = [];
27  ...
28
29  studentConcenrs::Int =0;
30  trace_studentConcerns::[Int] =[];
31
32  teachingQuality::Bool =0;
33  trace_teachingQuality::[Bool] =[];
34
35  eventTrace::[T] = []
36
37  → {
38     calender ← RegieterForLectureSlot(self);
39     calender ← RegieterForHour(self);
40     ...
41  };
42
43  // IncomingEvents
44  StudentQuery → { ... };
45  Complain → { ... };
46
47  //InternalEvents
48  TeachingPreparation → { ... };
49
50  //TimeEvents
51  Hour→ {
52   variable_PreprateForLecture ← Hour;
53   eventTrace:= eventTrace + [Hour]
54   //delegate events to all relevant inner actors
55  };
56  Day(day::Int)→ { ... };
57  Month(month::Int) → { ... };
58  LectureSlot( ... ) → { ... };
59  DeliverLecture( ... ) → { ... };
60  ActionDone( ... ) → { ... };
61
62  variableActPrepareForLecture::ActPrepareForLecture = new actPrepareForLecture();
63  act actPrepareForLecture()::ActPrepareForLecture { ... }
64  //For all other Actions
65   LeverIncreaseTeachingPreparation → { propensityOfTeachingPreparation = [1,5]  }
66 };
```

Figure 5.22 Overview of translated ESL specification

- Translated ESL functions that computes the OrgML `Functions` (line 5 and 6).
- Functions to compute OrgML `Measures` (line 8 and 10).
- Translated `Events` specification (line 42–60).
- Specification to realise OrgML `Actions` (line 62 and 63).

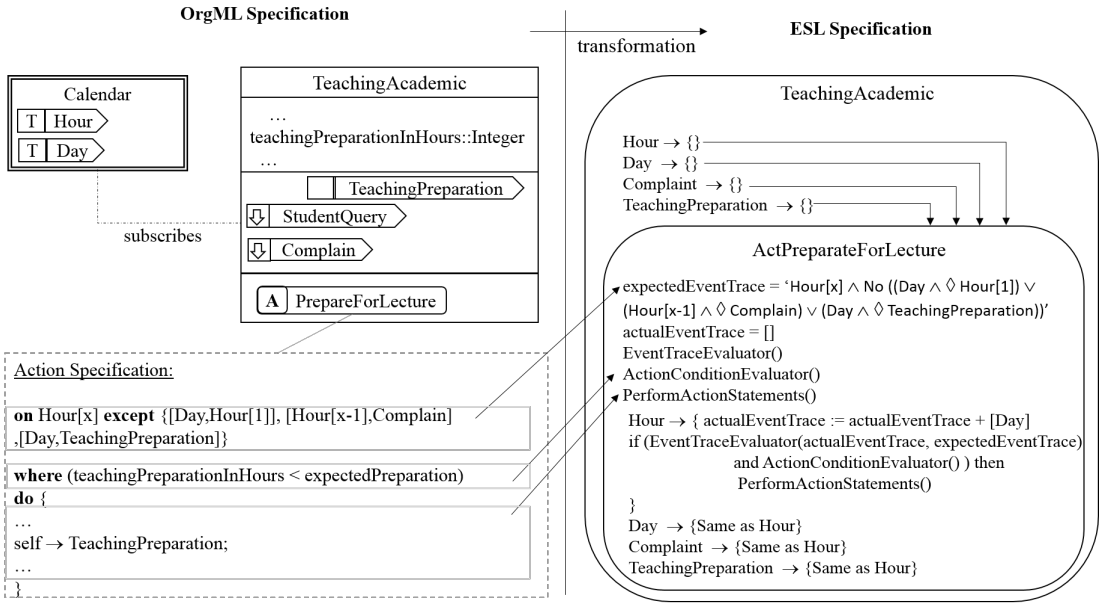


Figure 5.23 Overview of Action transformation

The translation of an OrgML `Action` to an equivalent ESL specification and transformation of OrgML `Calendar` to ESL actor require complex transformation. They are discussed below:

Action Translation: A translation strategy is pictorially illustrated using '*PrepareForLecture*' `Action` of *TeachingAcademic* `OrgUnit` in Figure 5.23. As shown in the figure, '*PrepareForLecture*' `Action` performs a set of `Statements` when the following two conditions are satisfied:–

- A pattern matching over event trace: Every '*Hour*' `TimeEvent` (specified as all occurrences of '*Hour*' using `Hour[x]`) except it is first the '*Hour*' of a '*Day*' (specified as '*Day*' followed by the first occurrence of '*Hour*' using `[Day,Hour[1]]`) or a '*Complaint*' is raised in the previous '*Hour*' (specified using `[Hour[x-1], Complaint]`) or '*TeachingPrepration*' is already raised on the same '*Day*' (specified using `[Day, TeachingPreparation]`). The condition can be translated to LTL formula as shown in Figure 5.23 to use the pattern matching over event traces.
- Condition over state variables: '*teachingPreparationInHours*' is less than expected preparation value.

The `Action` of '*TeachingAcademic*' `OrgUnit` is translated into an inner ESL actor (named as '*ActPreparateForLecture*') of '*TeachingAcademic*' ESL actor. The translated ESL specification realises the '*PrepareForLecture*' `Action` as follows:

```
//An OrgML Calendar specification
 Calendar {
   Hour= primitive
   Day= Hour[8]
   Week =Day[5]
   Month= Day[30]
   LectureSlot = [ (Hour(3) of Day(2) of Week), (Hour(5) of Day(4) of Week except Day(4) of
      Month)]
 }
```

```

 // Translated ESL Actor specification
 act calendar()::Calendar {
  hour::Int = 0;
  hourSubscriber::[T] = [];
  ...
  Hour→ {
    hour:= hour + 1;
    for n::Int in 0..(length(hourSubscriber)-1) do nth(hourSubscriber,n) ← Hour
  };
  ...
  Time(primitive::Int)→ {
      self ← Hour;
      if ((hour % 8) = 0) then self ← Day else {};
      if ((day % 5) = 0) then self ← Week else {};
      if ((day % 30) = 0) then self ← Month else {};
      if ((((hour % 8) = 3) and ((day  % 5) = 2)) or (((hour % 8) = 5) and ((day  % 5) = 4)
     and not((day % 30) = 4) ) ) then self ← LectureSlot else {}
  };
```

Figure 5.24 Illustration of Calendar

- Each `Action` is translated into an inner actor with the following elements – (i) an ESL variable to represent expected event trace (*i.e.*, '*expectedEventTrace*'), (ii) an ESL variable to capture actual event trace (*i.e.*, '*actualEventTrace*'), an ESL function to evaluate the pattern matching of actual event trace with respect to the expected event trace (*i.e.*, '*EventTraceEvaluator()*'), a function to evaluate the state variables (*i.e.*, '*ActionConditionEvaluator()*'), and a function to perform the statements specified as action statement (*i.e.*, '*PerformActionStatements()*').

- Expected event trace is formulated based on the semantic interpretation of the event specification (as specified in Figure 5.15).

- All relevant events of the outer actor, such as '*Hour*', '*Day*', '*Complaint*' and '*TeachingPreparation*', are delegated to inner actor (so that the inner actor can trace the events).

- Event definitions of the inner actor update `Variable` that represents actual event trace. The translated event definition triggers the action statements when event pattern and state variable conditions are satisfied.

The transformation rule to transform OrgML `Action` and their `Statements` to ESL specification are presented in Appendix C.2.1.

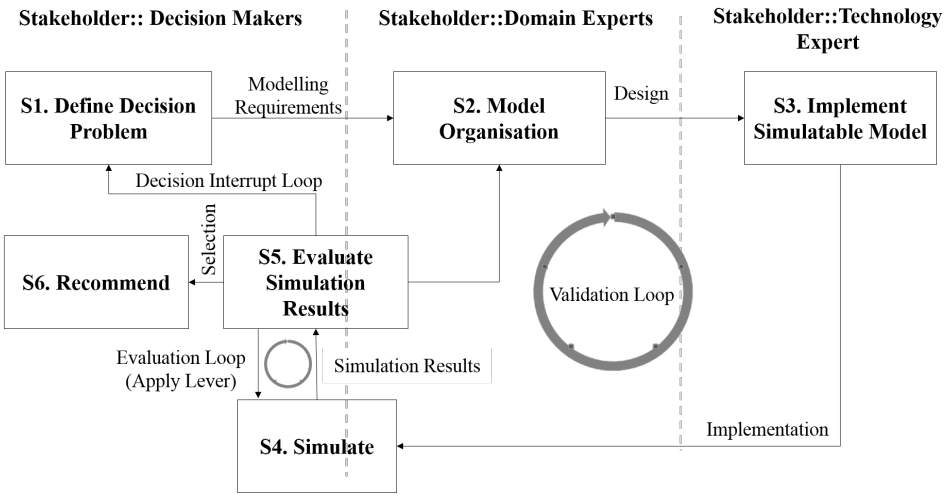


Figure 5.25 Method for model construction, validation and decision-making

Calendar Translation: An OrgML `Calendar` is translated into ESL actor and the contained `TimeEvents` are translated into ESL events as shown in Figure 5.24. Line 1 – 8 presents a definition of OrgML `Calendar` that specifies five `TimeEvents`: '*Hour*', '*Day*', '*Week*', '*Month*' and '*LectureSlot*'. The `TimeEvent`'*Hour*' is mapped to a primitive event and rest of the `TimeEvents` are expressed with respect to '*Hour*' and other `TimeEvents`. The ESL actor specification along with the logic for deriving non-primitive `TimeEvents` and sending those `TimeEvents` to all subscribed actors are listed in line 10 – 26. The transformation rule to transform OrgML `Calendar` to ESL actor specification is presented in Appendix C.2.2.

5.6 Method

An integrated and iterative method is introduced to represent necessary aspects of an organisation as an OrgML model, ascertain model validity and simulate/execute constructed model for required *what-if* analysis. The proposed method contains six steps over three swimlanes or types of responsible stakeholders as shown in Figure 5.25. Three stakeholders are – *decision makers*, *domain experts* and technology experts. The decision-makers are the users or *clients* [166] of a simulation activity, *domain expert* is a team that contains the experts from the problem domain and *modellers* [166] (*e.g.*, OrgML modeller) and the *technology experts* are the programmers who can encode the conceptual model into computerised model (*e.g.*, ESL programmers).

The method steps are: *Define Decision Problem* [S1], *Conceptualisation of Organisation Model* [S2], *Implement Simulatable Model* [S3], *Simulation* [S4], *Evaluation of Simulation*

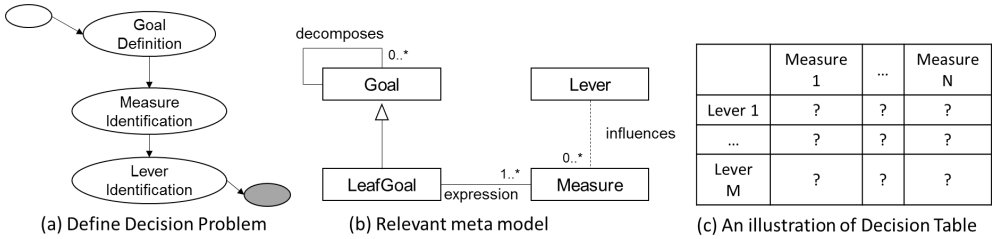


Figure 5.26 Modelling artifacts of Define Decision Problem process step [S1]

Results [S5], and *Recommendation* [S6]. Step S1 formalises the decision problem and defines the scope for *what-if* scenario playing by specifying the `Goals`, `Measures` and `Levers` of an `Organisation`. The step S2 conceptualises a model that imitates a real organisation for the purpose of a decision problem defined in step S1. The step S3 transforms the conceptual model into a simulatable model. Step S4 simulates the scenario defined in step S1. Step S5 evaluates the simulation results with the step S6 providing the recommendations.

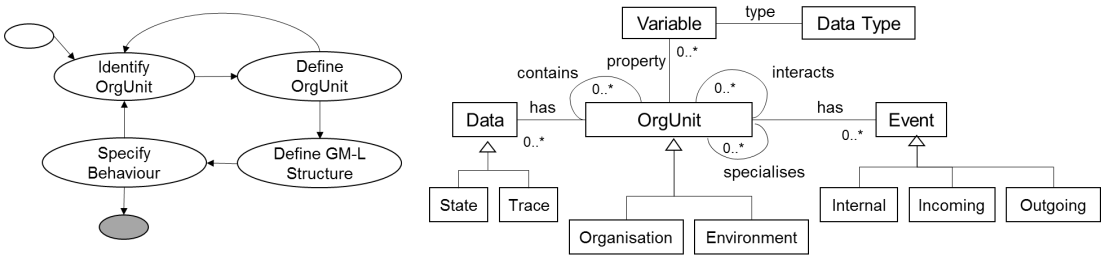
The method considers three representations: *problem entity*, *conceptual model* and *computerised model*, refines the two-step model construction process, and adopts *operational validity* to ascertain the model validity as recommended by Robert Sargent in [174]. In particular, the process steps S1 and S2 of the proposed method capture the conceptual model of a decision problem, the process step S3 converts captured conceptual model into simulatable model, and the loop containing the process steps S2, S3, S4 and S5 realises the *operational validity*.

The management viewpoints of organisational decision-making process are also considered as discussed in section 5.2.1. The process step S1 realises the activity *Recognition of Decision Requirement*, process step S2 realises the activity *Diagnosis and Analysis of Causes* and *Development of Alternatives*, the loop S4 and S5 with the process step S6 realise the activity *Selection of Desired Alternatives* of the organisational decision-making process recommended by Richard Daft in [70]. In addition, the concept of *decision interrupts* [123] to explore the decision alternatives that emerge while evaluating the known decision alternatives is realised through the loop S5, S1, S2, S3, and S4 as shown in Figure 5.25. Detailed activities of the proposed process steps are illustrated below:

Define Decision Problem [S1]: This step identifies the `Goals`, `Measures` and `Levers` from a problem entity using three sub-steps: *Goal Definition*, *Measure Identification* and *Lever Identification* as shown in Figure 5.26 (a).

Table 5.3 An illustration of Decision Table

	Student Concern	**High Quality Lecture**	**High Quality Publication**	**Research Earning**	**Research Collaboration**
Increase Teaching Preparation Hours	?	?	?	?	?
Balance of Teaching and Research activities	?	?	?	?	?
Introduce Better Timetable	?	?	?	?	?



(a) Construction of conceptual model

(b) Relevant meta model for Conceptualisation of Organisation process step

Figure 5.27 Modelling artifacts of Conceptualisation of Organisation Model process step [S2]

The *Goal Definition* sub-step uses a top-down approach to define `Goals` and goal decomposition structure, *Measure Identification* sub-step identifies `Measures` for all `LeafGoals` of the constructed goal model, and sub-step *Lever Identification* identifies a set of `Levers` that may influence the `Values` of identified `Measures`. The goal decomposition structure along with the goal-measure relationship of ABC University is shown in Figure 5.18 and an example of `Lever` specification is illustrated in Figure 5.20.

The process step S1 creates two artifacts. The primary artifact is an instance of a part of OrgML meta-model that describes GM–L structure, *i.e.*, the `Goal` – `Measure` relationships and a list of `Levers` that may influence the `Measures`. The relevant part of the OrgML meta-model that captures the GM-L structure is depicted in Figure 5.26 (b). The second artifact is a derived artifact from the GM-L structure that explicates the *what-if* scenario in the form of a table, termed as *decision table*. A decision table can be constructed by considering the identified `Levers` as rows and `Measures` as columns as shown in Figure 5.26 (c). In a decision table, each grid is a question that explores the expected/possible value of the `Measure` when a `Lever` is applied to the organisational model. A decision table of the `Goal` – `Measure` structure shown in Figure 5.18 along with a set of University specific `Levers` are shown in Table 5.3.

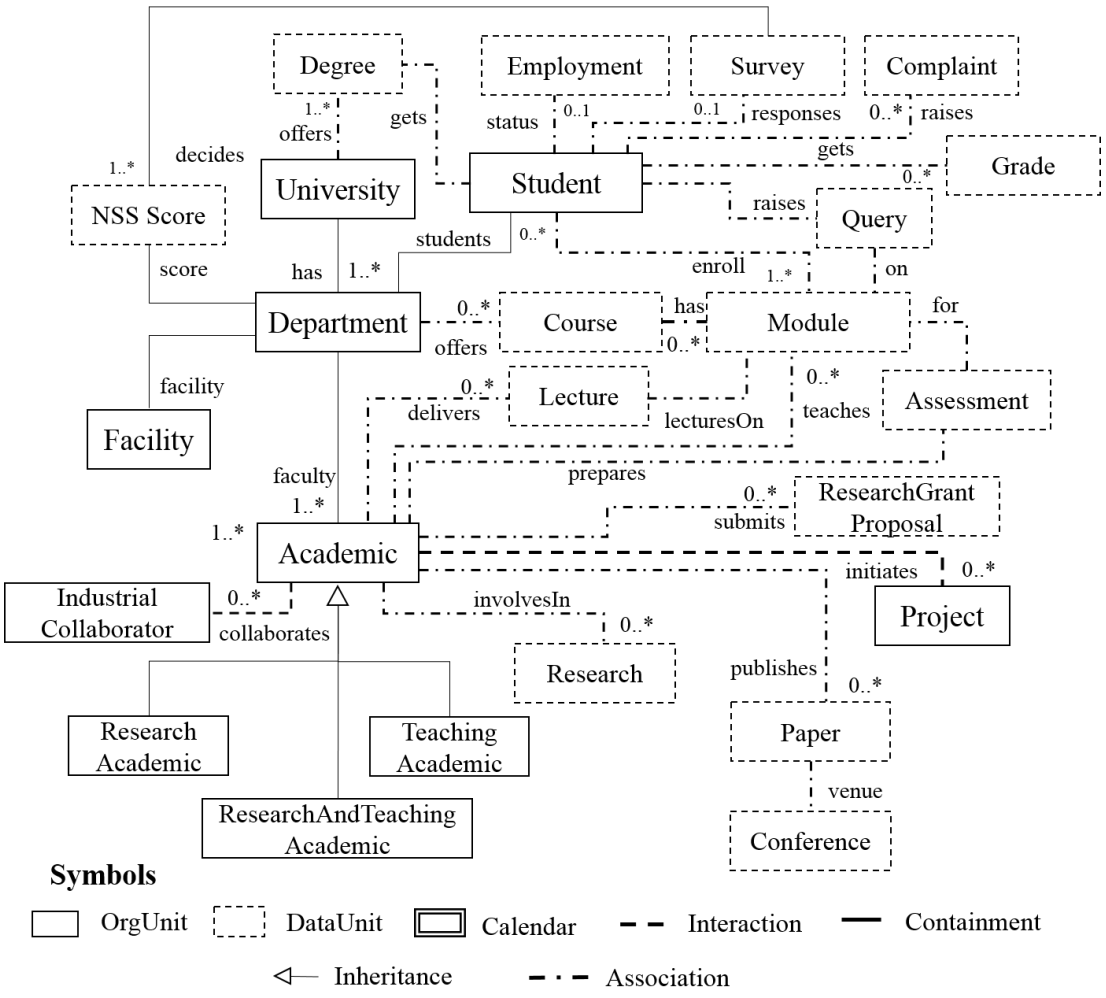


Figure 5.28 Examples of OrgUnits and DataUnits of ABC University

Conceptualisation of Organisation Model [S2]: This step conceptualises a purposive model of the `Organisation` by considering derived GM–L structure as the purpose for organisation modelling. In particular, this process step captures `Structure`, `Behaviour`, `Variables`, `Traces` and `Events` of an `Organisation` to understand `Measures` in the presence of `Levers` of a GM–L structure derived in process step S1. This process step performs four activities, as shown in Figure 5.27 (a), as described below:

1. **Identify OrgUnits** activity identifies the prospective `OrgUnits` and `DataUnits` from a problem entity. All key elements of a problem entity are identified and then they are classified into `OrgUnits` and `DataUnits` based on their characteristics. The elements which are active, *i.e.*, reactive, intentional and has autonomous behaviours, are classified as `OrgUnits`, whereas the elements which are collection of data are classified as `DataUnits`.

A set of `OrgUnits` and `DataUnits` of ABC University that are identified from the problem entity description illustrated in section 1.6 (of Chapter 1) are shown in Figure 5.28. The elements such as *University*, *Department*, *Academic*, *Student*, *Industrial Collaborator* are the examples of `OrgUnits`. The *Course*, *Module*, *Query*, *Complains*, and *Lectures* are the examples of the `DataUnits`.

2. **Define OrgUnit** activity chiefly defines the `OrgUnit` structure and identifies *containment*, *interaction* and *inheritance* relationships of the identified `OrgUnits`. In particular, this activity defines `Parameters` and `Variables` to represent `State` and `Trace` information, identifies `Events` that interact with other `OrgUnits`, and defines structural containments and the inheritance relationships. The relevant subset of OrgML meta-model that is instantiated in this activity is shown in Figure 5.27 (b). This activity also defines the `DataUnits` by identifying their `Variables`.

   Activities *Identify OrgUnit* and *Define OrgUnit* navigate the containment relationships to explore the decomposition and/or composition relationships of the `OrgUnits`, interaction relationship to explore the vertical interactions, and the inheritance relationship to explore specialisation (as shown in Figure 5.28).

3. **Define GM-L of OrgUnit** identifies the `Goals` that an OrgUnit owns, the `Measures` that it can produce, and the `Levers` that can be applied on it. For an example, the `Goal`, `Measure` and `Lever` of *TeachingAcademic* is shown in Figure 5.21.

4. **Specify Behaviour** captures the behavioural specification of the identified OrgUnits. This step specifies the `Deterministic`, `Stochastic`, `Temporal` and `Adaptive` behaviour of `IncomingEvents`, `InternalEvent`, subscribed `TimeEvent`, `Functions` and `Actions`.

Implement Simulation Model [S3]: This process step translates a conceptual model defined using OrgML into ESL specification. It uses the rules defined in Table 5.2. The input of this step is the OrgML models and output is an ESL specification. For example, an OrgML model of a *Teaching Academic* as shown in Figure 5.21 is translated to an ESL specification as shown in Figure 5.22.

Simulation [S4]: This step runs the simulation model (with or without `Lever`), observes `Measures` from simulation runs, and captures results in a row of the decision table formulated

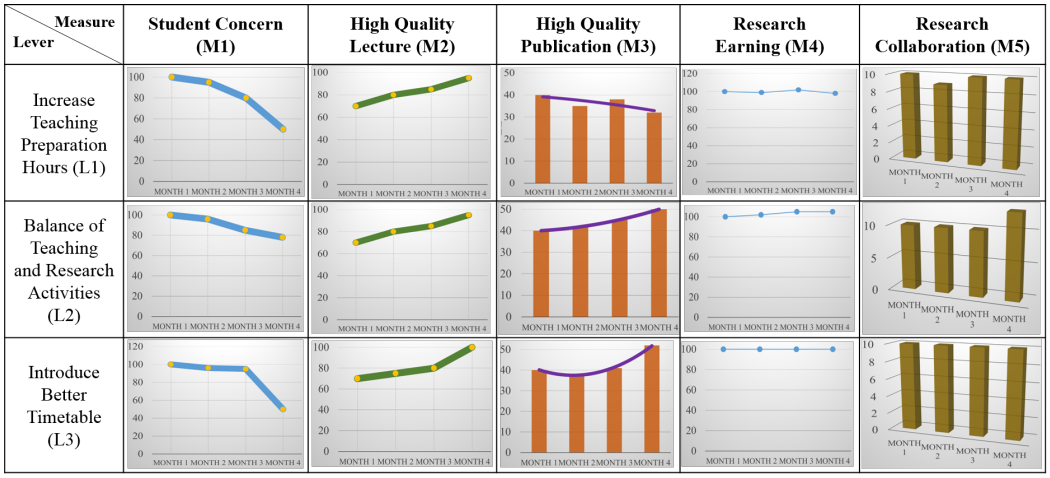


Figure 5.29 Illustrative outcome of what-if analysis

in process step S1. Each simulation run helps to answer the *what-if* questions of a decision table row. This research uses an ESL based simulation to analyse *what-if* scenario constructed in the form of decision table, and visual representations of the `Measures` and `Traces` to understand the consequence of the `Levers`.

An illustration of the simulation based *what-if* analysis outcomes captured using decision table is shown in Figure 5.29. The graph of the each cell depicts a trend of a specific `Measure` in the presence of a `Lever` over time axis. The time axis indicates the near term and long term consequences of a `Lever`. For example, the graph in cell of row L1 and column M1 shows the impact of `Lever` '*Increase Teaching Preparation Hours*' on the `Measure` '*Student Concern*' over time. Each point in the graph represents the sum of the complaints received and query raised to all academics in a '*Month*' (as defined in Figure 5.21).

Evaluation of Simulation Results [S5]: This step evaluates the simulation results captured in the decision table. Human experts interpret the simulation results by triggering one of the following possibilities: (i) initiate a *Validation Loop* that iterates process steps S2–S3–S4–S5 in case simulation results of a known scenario don't match the expected outcome (i.e., operation validity is not satisfied), (ii) explore next `Lever` of a decision table by triggering an *Evaluation Loop* that iterates process steps S5 and S4 , (iii) select the best possible `Lever` once all levers are evaluated through simulation (*i.e.*, S5 to S6 transition), (iv) identify a new `Lever` *i.e.*, add a new entry in the decision table and reiterate the overall process using *Decision Interrupt Loop* described in Figure 5.25. Four scenarios are illustrated using the graphs shown in Figure 5.29 as follows:

- **Model Validation**: Consider a `Department` where the impact of `Lever` '*Increase Teaching Preparation Hours*' on `Measure` '*High Quality Lecture*' is known (from historical data). But a simulation result is showing a different outcome then there is a high possibility that the constructed model is inaccurate for the `Levers`. The constructed model is typically validated using sufficient number of known scenarios[6].

- **Lever Exploration**: All cells of the constructed decision table need to be populated using an iterative *what-if* analyses where each iteration considers a `Lever`. The iterations terminates when all cells are populated in a decision table. For example, a decision table as shown in Figure 5.29.

- **Decision Interrupt**: Decision makers may (manually) interpret results captured in a decision table and decide to explore more `Levers` as part of decision space exploration. For example, decision maker may choose to explore an option that combines the `Lever` L1 and L2 as new `Lever` or a new set of `Goals` and thus `Measures` can be introduced as part of *Decision Interrupt Loop*.

- **Proceed to Recommendation**: This is a situation when all above loops are sufficiently concluded, *i.e.* all `Levers` of a decision table are explored on a validated model and no new `Lever` is identified while exploring identified `Levers`.

Recommendation [S6]: This step recommends one or more `Levers` that can be implemented in real organisation. Decision makers take the decision by evaluating quantitative simulation results (or evidences that indicate near term and long term consequences) as shown in Figure 5.29. This research argues that a decision based on a populated decision table is quantitatively justified. Moreover, such decision can be considered as informed decision as the near-term and long-term consequences are known to the decision makers.

Model and simulation validation

The proposed method considers two kinds of validations – (i) structural validation and (ii) operational validation. The structural validation primarily ensures the OrgML model defined in process step S2 conforms to the GM–L structure defined in process step S1. Two principal rules of structural validation are:

[6]The required number of *what-if* analyses to ascertain the model validity is a discretion of the involved decision makers

1. Measure well-formedness: All `Measures` of the GM–L structure identified in step S1 must be measured by an identified `OrgUnit`.

2. Lever well-formedness: All `Levers` of GM–L structure identified in step S1 must be (jointly or individually) owned by identified `OrgUnits`.

The operational validity is ensured through a validation loop that iterates over process steps S5, S2, S3 and S4 and compares experimental results with real or predicted data. The validation process uses the operational graphics, *i.e.*, graphical and/or tabular representation of the `Measures` as a basis for the evaluation, and rely on human experts to certify the validity as described in section 5.2.1. The other validation techniques, such as *data validity* or *conceptual validity*, while being effort and time intensive, provide no additional certainty as discussed in [70].

5.7 Summary

An actor-based behavioral simulation aid is presented in this chapter. The proposed approach is illustrated using three research artifacts - (i) the OrgML meta-model that serves as a domain specific specification for organisational decision-making, *i.e.*, *Contribution 2*, (ii) a transformation strategy to convert the OrgML specification into an ESL based simulatable specification, *i.e.*, *Contribution 3*, (iii) a method as a guidance to construct conceptual model, transform conceptual model into simulatable model, ensure the validity of the constructed models, and perform the required *what-if* scenario in a systematic manner, *i.e.*, *Contribution 4*. Fundamentally, the proposed approach uses the *modelling and simulation* as the philosophical basis, adopts an actor based modelling abstraction and the bottom-up simulation as a technological basis, considers the methodological rigour used by simulationist to raise the epistemic value of the proposed simulation, and correlates with the management viewpoints to introduce expected management rigour.

From the utility perspective, the OrgML meta-model is presented as an aid to capture the decision-making related requirements using GM–L structure and model complex socio-technical organisations using a set of composable and interacting OrgUnits. The use of ESL and the proposed OrgML to ESL transformation strategy are effective enabler of the bottom-up simulation approach. The proposed method is highlighted as methodological framework to support technology aided evidence driven organisational decision-making.

Chapter 6

Proof of Concept Technology Aids

This research considers three technology aids to approach organisational decision-making using the proposed OrgML based approach. The technology aids are: (i) a domain specific language to capture necessary information of an organisation using a set of OrgUnits, (ii) simulation technology for *what-if* analysis, and (iii) a visualisation aid to represent simulation results in an intuitive form for sense-making. Two of the technology aids: *OrgML workbench* as an integrated development environment (IDE) for OrgML spacification and *OrgViz Data Visualiser* as a visualisation aid are developed and the ESL technology (*i.e.*, ESL editor and simulation engine) is used as simulation technology.

This chapter presents an overview of the proof-of-concept technology implementations. Section 6.1 highlights critical and effort intensive activities of the proposed organisational decision-making approach. Implementation details of OrgML workbench is presented in section 6.2. Design considerations and implementation details of OrgViz Data Visualiser is discussed in section 6.3. A framework, termed as *OrgDM framework*, is designed to integrate the technology aids in an effective and systematic manner. OrgDM framework is presented in section 6.4. From methodological perspective this chapter focuses on *Instantiation* of the research artifacts produced in *Conceptualization of proposed solution* research activity (of Figure 2.4).

6.1 Core activities and expected technology aids

The approach for evidence-driven organsational decision-making presented in Chapter 5 involves three broad activities: modelling, simulation and decision-space exploration as summarised

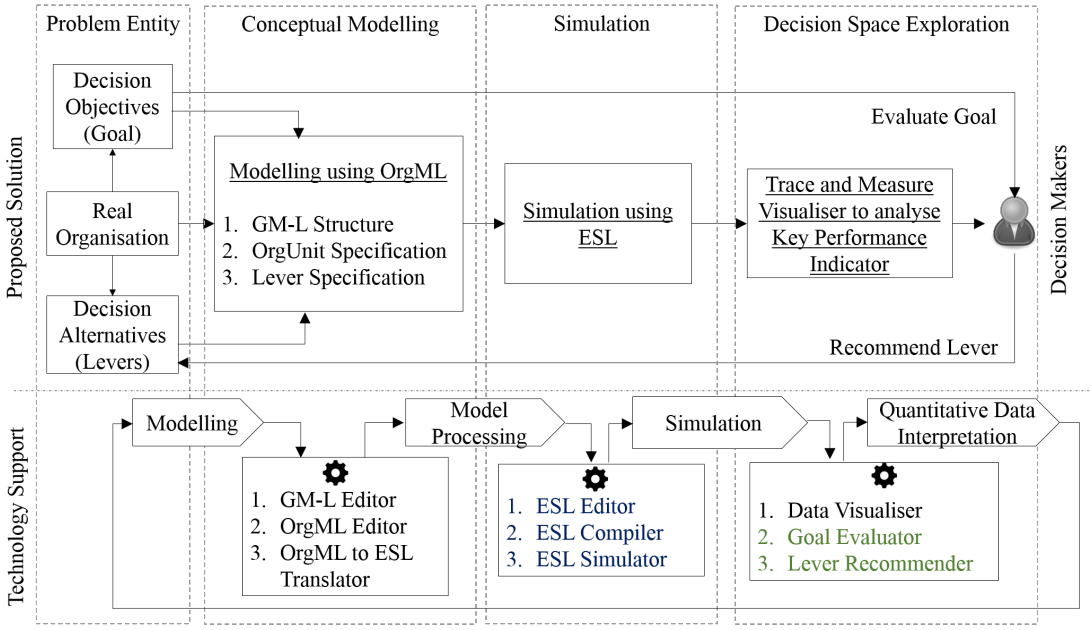


Figure 6.1 Core activities and expected technological aids

in Figure 6.1. The modelling activity captures GM-L structure and organisation specification using OrgUnits. It also constructs a simulation model by translating OrgML specification. The simulation activity produces `Measures` and `Traces` for constructed model with or without `Levers`. Activity decision space exploration interprets numerical data obtained from simulation runs and evaluates them with respect to organisational goal to decide an option from following three alternatives – (i) fine tune an existing `Lever` (*e.g.*, explore parametric values of a lever) (ii) combine existing `Levers` for better outcome and (iii) explore a new `Lever` prior to a recommendation.

Therefore, the technology aids that help the organisational decision-making approach presented in the earlier chapter are:

1. GM-L editor to capture GM-L structure.
2. OrgML editor to capture organisational model using OrgML specification and establish consistency within organisational model and between GM-L and organisational model.
3. OrgML to ESL translator for simulation based *what-if* analysis.
4. ESL editor for simulation specification.
5. ESL simulator for simulation.

6. Data visualiser to visualise simulation data in an intuitive manner. Two modes of visualisation are useful for human-centric interpretation – (i) visualise a simulation run and (ii) visualise historical simulation results.

7. Data analytics capability to help decision makers in decision space exploration.

In this research, the *OrgML workbench* realises GM-L editor, OrgML editor and OrgML to ESL translator, the *OrgViz Data Visualiser* is developed to support the needs of data visualisation capabilities, and a modelling and simulation framework, *OrgDM* is conceptualised to integrate ESL technology. The rest of this chapter discusses the implementation details of *OrgML workbench*, *OrgViz Data Visualiser* and *OrgDM framework*.

6.2 OrgML Workbench

The OrgML workbench is a domain-specific *language workbench* [79] that conforms to OrgML meta-model (presented in Figure 5.9). It supports a set of language editing features for two interoperable languages termed as *GM-L specification* and *Organisation specification*. The GM-L specification language is conceptualised to help decision makers capture GM-L structure using a simple and intuitive form. The Organisation specification language is designed for domain experts to capture the necessary aspects and characteristics of an organisation. The expressiveness is a key characteristic of Organisation specification language. The interoperability between two specification languages is expected to ensure structural and conceptual consistency as they collectively specify the necessary information for organisational decision-making. The concrete syntaxes of the supported languages, language workbench related features and an implementation of OrgML workbench are discussed in this section.

6.2.1 Language definitions

The GM-L specification and Organisation specification languages are designed by considering parts of OrgML meta-model as abstract syntaxes. The GM-L specification language conforms to `Goal`, `Measure`, `Lever` definitions and their relationships, whereas the Organisation specification language conforms to the part of OrgML meta-model elements that describe `OrgUnit`, `DataUnit` and `Calendar`. The key elements of the textual concrete syntax of GM-L specification and Organisation specification are presented in this subsection.

```
0  /*        OrgML Specification Keywords        ,    OrgML Meta Elements    */
1  gml ::= GML {                                        GML Specification
2    goals    : (goal*)                                 Goal Spec
3    measures : (measure*)                              Measure Spec
4    levers   : (lever*)                                Lever Spec
5  }
6  goal ::= id [ description ] g_expr                    Goal Declaration
7
8  g_expr ::= { g_expr g_reln g_expr }                   Goal Decomposition
9    | ⇒  leaf_goal                                      LeafGoal
10
11 g_reln  ::=   ;                                       And Relation
12   | |                                                 Or Relation
13   |   →                                               Sequence Relation
14
15 leaf_goal ::= m_exp                                   Quantitative Expression
16   | r_exp                                             Relative Expression
17
18 m_exp ::= measure                                     Measure
19   |  consts                                           Constants
20   |  exp op exp                                       Binary expression
21   |  Not exp                                          Negation
22   |  fun(exp*)                                        Function Call
23   |  [ exp* ]                                         List Of Expressions
24   |  []                                               Empty Expression
25
26 r_exp ::= [prefix] qualifier [suffix]                 Relative Expression
27
28 prefix ::= Always | Never
29 qualifier ::= Increase | Decrease                     Relative Operations
30   | Maintain | Maximise | Minimise
31 suffix ::= t_exp time                                 Time Expression
32 t_exp ::= At | Before | After | During
33
34 measure ::= id                                        Measure Declaration
35 leaver ::= id                                         Lever Declaration
36 time :: id                                            TimeEvent
```

Figure 6.2 Syntax of GML specification

GM-L specification language

The proposed GM-L specification focuses on the model elements which are considered to be instantiated in process step *Define Decision Problem* [S1] of the organisational decision-making method presented in Chapter 5. The model elements include `Goal`, `Goal` decomposition, `Measure` and `Lever` of OrgML meta-model. A concrete syntax that highlights the core concepts of GM-L specification language is shown in Figure 6.2. As shown in the figure, the GML specification contains *goals*, *measures* and *levers* specifications (line 1–5). A *goal* can either be decomposed in finer goals (as shown in line 8) or it can be mapped to a *measure* to indicate a `LeafGoal` (as shown in line 9). The decomposition relationships (*i.e.*, *g_reln*) can be specified using one of the three goal decomposition relations: *and*, *or* and *sequence* (as shown in line 11–13). The `LeafGoal` to `Measure` mapping can be specified either through a quantitative expression (*i.e.*, *m_exp*) or relative expression (*i.e.*, *r_exp*). The quantitative expressions are primarily mathematical and logical operators over measures as shown in line 18–24, whereas the relative expression describes expected value of a *measure* with respect to its instances. A set of language constructs such as *Increase*, *Decrease*, *Maintain*, *Maximise* and *Minimise* along

```
1  GML ABCUniversity {
2      goals:
3          Goal ImproveRanking [Improve University Ranking]        {
4              ImproveResearchQuality ; ImproveTeachingQuality
5          };
6          Goal ImproveResearchQuality => [  YearlyPublications > 100 ];
7          Goal ImproveTeachingQuality => [  Always Minimise YearlyStudentComplaints ];
8
9      measures:
10         Measure YearlyPublications;
11         Measure YearlyStudentComplaints;
12
13     levers:
14         Lever BalanceResearchAndTeaching;
15         Lever RecuitResearchers;
16 };
```

Figure 6.3 An illustration of textual GM-L specification

with suitable prefix (such as *Always* and *Never*) and suffix that include a time expression are proposed to specify the relative expression.

The measures and levers are defined as simple labels as shown in line 34 and 35. They are expected to be introduced in GM-L specification and explicitly specified in Organisational specification.

An illustration of GM-L specification is presented using a GM-L structure of ABC University in Figure 6.3. The illustration considers a goal that aims to increase its ranking by improving research quality and teaching quality where the research quality can be measured using yearly publication counts and the teaching quality can be measured using the number of student complaints. The primary goal of ABC University is represented using '*ImproveRanking*', which is decomposed into two leaf level goals: '*ImproveResearchQuality*' and '*ImproveTeachingQuality*' using an '*and*' decomposition relationship. The leaf goal '*ImproveResearchQuality*' is mapped to '*YearlyPublications*' measure using a quantitative expression (*i.e.*, '*YearlyPublications*' should be more than 100) whereas leaf goal '*ImproveTeachingQuality*' is mapped to '*YearlyStudentComplaints*' using a relative expression (*i.e.* value of '*YearlyStudentComplaints*' always should be in decreasing order). The specification also introduces two levers: '*BalanceReasearchAndTeaching*' and '*RecruitResearchers*'.

Organisation specification language

Organisation specification is defined to capture necessary information of an organisation as proposed in *Model Organisation* [S2] process step of organisational decision-making method (presented in Chapter 5). It specifies a set of interacting `OrgUnits`, a collection of `DataUnits`

```
0  /*        OrgML Specification Keywords        ,    OrgML Meta Elements    */
1  model ::= import_stmt calendar { element*}                  OmgML Spec
2
3  import_stmt ::= import ( orgml_spec_name* )                 Import OrgML Spec
4  calendar::= Calendar   id { time* }                         Calendar  Entity
5  element ::= data_unit | org_unit | function                 Element Types
6
7  data_unit ::= DataUnit   id  { ( variable*) }               DataUnit  Declaration
8
9  org_unit ::= OrgUnit                                        OrgUnit  Declaration
10  [extends org_unit_name] {                                  Inheritance
11   goals: (goal*)                                            Goal Specifications
12   measures: (measure*)                                      Measure Declarations
13   variables:  (property*)                                   Variable & Trace Declarations
14   uses: ( id*)                                              Variables from other  OrgUnit
15   subscribes     : (time_event_name*)                       Subscribed TimeEvent
16   consumes   : (behavioural_event*)  [trace]                IncomingEvent
17   produces    :  (outgoing_event*)  [trace]                 OutgoingEvent
18   internal-events:  (behavioural_event* ) [trace]           InternalEvent
19   functions: (function*)                                    Function Specifications
20   actions: (action*)                                        Action Specifications
21   levers    :  (lever*)                                     Lever  Specifications
22  }
23
24 property ::= [(@ augmentation)*] variable                   Encapsulated & Exposed Variables
25 variable::=  id :: type [ := exp ]                          Variable
26
27 type::= data_unit_name                                      User defined  DataUnit
28  |  org_unit_name                                           User defined  OrgUnit
29  |  Integer | String | Double
30  |  Date | Boolean                                          Primitive Type
31  |  [ type ]                                                List
32
33 augmentation::= export | parameter                          Exposed Variable or Parameter
34  | trace ( time_event_name)                                 Trace Variable
35
36 measure::= variable @ time_event_name
37   [Display chart_type meta_data]                            Visualisation Mechanism
38
39 chart_type::= BarChart | PieChart | Line
40  | BubbleChart | Table                                     Chart Type
41
42 behavioural_event::=
43  id (parameter* ) → { stmt*}                               Behavioural Event
44
45 outgoing_event::= id (parameter* )                         OutgoingEvent
46
47 parameter ::= id type
48
49 lever ::=      Lever     id (lever_spec* )                  Lever  Declaration
50 lever_spec::= At event Apply { lever_stmt* }               Lever  Spec
51 lever_stmt::= variable_name := exp                         Variable assignment
52  |  Replace p_event By p_event                             Event Replacement
53  |  Ignore p_event                                         Ignore  an Event
54  |  Deactivate action                                      Deactivate  an Action
55  |  Omit outgoing_event_name                               Don't send an OutgoingEvent
```

Figure 6.4 Organisation specification language syntax

and a `Calendar` entity. A concrete syntax of core constructs of Organisation specification language is shown in Figure 6.4.

As shown in the figure, Organisation specification language contains three sections: import, calendar and element description sections. The import section `imports` a set of OrgML files that contain GM-L specification and other Organisation specifications. The calendar section defines the `Calendar` entity of OrgML meta-model. The construct *calendar* is defined using a set of '*time*' constructs (line 4) where the '*time*' construct is a textual specification of OrgML `TimeEvents` as defined in Figure 5.14. Element description section defines OrgML `Functions`, `DataUnits` and `OrgUnits` using the terms *function*, *data_unit* and *org_unit* respectively as

shown in line 5. The concrete syntax of *function* considers the syntax definition of OrgML `Function` as defined in Figure 5.14 (presented in Chapter 5).

Concept `DataUnit` of OrgML meta-model that contains a set of `Variables` can be specified using a term *data_unit*. The term *data_unit* (shown in line 7) contains a set of *variables* where variable is a typed element that represents the OrgML `Variables`. A *type* can be one of the three alternatives – (i) primitive type, such as *Integer*, *String*, *Boolean*, *Double* and *Date* as shown in line 29–30, (ii) a list as shown in line 31, or (iii) an user defined type, such as `DataUnit` and `OrgUnit` definitions, as shown in line 27–28.

A concrete syntax to specify `OrgUnit` concepts[1] along with the inheritance relationship is shown in line 9 – 22 of Figure 6.4. The syntax of a `OrgUnit` declaration is shown in line 9, line 10 captures the inheritance relationship, line 11 specifies the `OrgUnit` specific goals, and measures can be specified using a syntax defined in 12.

A detailed syntax of measure specification is shown in lines 36–40. The measures are realised as variables that needs to be displayed at specific time interval using a suitable visualisation mechanism or *chart_type*. Supported *chart_type* are : *bar chart*, *pie chart*, *line*, *bubble chart* and *table* (an extensible list of options). In addition, a set of display properties can be supplied to the display unit through *meta_data* information (a set of name value pair to capture property names and their values).

The OrgUnit `Variables` can be declared using a syntax definition as depicted in line numbers 13, 24–34. Term *property* defines the syntax for OrgML `Variable`, exposed `Variables`, `Parameter` and `Trace`. Exposed `Variables` can be specified by augmenting a variable with '*export*' keyword (as shown in line 24, 25 and 33), `Parameter` can be indicated by augmenting a variable with a '*parameter*' keyword, and a trace can be declared by augmenting a variable with '*trace*' keyword along with a time event (as shown in line 34). The exposed variables of other `OrgUnit` can be accessed in an `OrgUnit` by declaring them as *uses* variable as shown in line 14.

The syntax of OrgML `Event` specifications that include the `TimeEvent`, `IncomingEvent`, `OutgoingEvent` and `InternalEvent` are specified as follows:

[1] An `OrgUnit` (of OrgML meta-model) contains `Variables`, `Functions`, `InternalEvents` and `Actions`. It exposes a set of `Variables`, uses `Variables` from other `OrgUnits`, consumes a set of `IncomingEvents`, produces `OutgoingEvents` and subscribes a set of `TimeEvents`. In addition, each `OrgUnit` has its own `Goals`, shows a set of `Measures` and capable of introducing a set of `Levers`

gml.orgml | Academics.orgml | ErrorExample.orgml

```
1  Domain Specification  ABCUniversity {
2  import gml ABCUniversity;
3  Calendar   { Hour = Primitive;Day= Hour[8];Month = Day[30]; Week = Day[5]; Year =  Month[12];
4      LectureSlot = [ Hour[2] Of Day[1] Of Week, Hour[5] Of Day[5] Of Week];}
5  DataUnit Module {
6          moduleName:: String;
7          credit::Integer := 4;
8          lectureSlot::Integer [] := [2,4,5];
9          lecturePrepTime::Integer := 10;
10 }
11 OrgUnit Academic
12 {
13     goals:
14         Goal ImproveUniversityRanking [Improve University ranking ] {};
15     measures:
16         TeachingStatistics = teachingStatistics @Month [Display BarChart];
17         TeachingStatisticsPie = teachingStatistics @Day [Display PieChart];
18         ComplaintsAndQueriesStatistics = complaintsAndQueriesStatistics @Month [Display BarChart];
19     variables:
20         @[parameter, export] academicName::String := "Academic1";
21         @parameter workingHours::Integer := 8;
22         @export workingExperience::Integer := 10;
23         @trace(Month) queryRaised::Integer := 0;
24         @trace(Month) compaintsReceived:: Integer := 0;
25         teachingStatistics:: Integer[][] := [[1,2, 3], [2,3, 4]];
26         complaintsAndQueriesStatistics:: Integer[][] := [[1,2, 3], [2,3, 4]];
27     subscribes: Hour, Day, Month, Year; //Subscribe Time Event
28
29     consumes:
30         @trace StudentQuery(String description) ;
31         @trace StudentComplaint(String severity,String description);
32     functions:
33         Function Boolean queryResolution(String[] queries) = {/* Logic to handle Queries */};
34         Function Boolean compiantResolution(String severity,String description) = {/* Logic */};
35 }
36 OrgUnit TeachingAcademic extends Academic
72 }
```

Figure 6.5 An illustration of textual OrgML specification

- Subscribe a set `TimeEvents` by creating a *subscribes* block and listing a set of *time* definitions as shown in line 15.
- Define `IncomingEvent` using a term *consumes*, a set of *behavioural_event* definitions and an optional *trace* indicator as shown in line 16.
- Declare `OutgoingEvent` using a term *produces*, a set of *outgoing_event* definitions and an optional *trace* indicator as shown in line 17.
- Define `InternalEvent` using a term *internal_events*, a set of *behavioural_event* definitions and an optional *trace* indicator as shown in line 18.

The *trace* indicator of the above definitions are used to indicate that an event needs be be traced, *i.e.*, the occurrence details should be captured along with the time stamp (*i.e.* OrgML `EInfo`). The syntax of *behavioural_event* includes an event name, a set of event parameters and a set of behavioural statements as shown in line 42–43. The syntax definition of behavioural

statements uses the term *stmt*, which is defined as part of `BSpec` syntax as shown in Figure 5.14. The *outgoing_event*, in contrast, declares an event by specifying an event name and list of event parameters as shown in line 45. An `OrgUnit` may contain `Functions` and `Actions`. They can be specified using `BSpec` syntax specification (shown in Figure 5.14) as highlighted in line 19 and 20.

Finally, the textual syntax of OrgML `Lever` specification is shown in line 21 and 49–55. As shown in line 49, a lever is a set of *lever_spec* (line 49) where each *lever_spec* is a tuple that contains an event and a collection of lever statements (*i.e.*, *lever_stmt*). Consistent with `Lever` specification proposed using variability modelling in Figure 5.19, a *lever_stmt* supports variable assignment, event replacement, ignore an incoming event, omit an outgoing event and deactivation of an action (as shown in line 51–55).

Figure 6.5 is an illustration of a textual specification of an OrgML model, which is pictorially shown in Figure 5.28 of Chapter 5. The specification imports a GM-L specification (line 2) and contains a calendar definition (in line 3–4), *Module* `DataUnit` definition (line no 5–10), *Academic* `OrgUnit` definition (line 11-35) and an extended `OrgUnit` that represents *TeachingAcademic* of ABC University.

Calendar specifies six `TimeEvents` where *Hour* is associated to the '*primitive*' time; `TimeEvents` *Day*, *Week*, *Month* and *Year* are specified using simple time expression; and *LectureSlot* is defined using complex time expression as shown in line 4. The definition of *LectureSlot* specifies two slots in a week: second hour of Monday and fifth hour of Thursday.

The *Module* `DataUnit` contains four variables with different variable types and assignment expressions. The definition of *Academic* `OrgUnit` contains *goals*, *measures* with various display mechanisms, and *variables* with appropriate augmentations to indicate parameters, exposed variables and traces as shown in line 13–24. A `Measure` definition specifies associated `Variable`, time interval and a visualisation means as follows:

```
TeachingStatistics = teachingStatistics @Month [Display BarChart]
```

The `Measure` definition '*TeachingStatistics*' indicates that the value of '*teachingStatistics*' `Variable` needs to be captured for every occurrence of '*Month*' `TimeEvent` and visualised using '*Bar Chart*'.

A `Variable` can be specified as `Parameter` of an `OrgUnit` using `@parameter` augmentation, can be exported from an `OrgUnit` using `@export` augmentation, and can be defined as a `Trace` variable using `@trace(a_time_event)` as shown in line 20–24. The value of a

gml.orgml | Academics.orgml | ErrorExample.orgml

```
11 OrgUnit Academic
36 OrgUnit TeachingAcademic extends Academic
37 {
38     goals:                                  //goal decomposition structure
39         Goal ImproveStudentSatisfection [Improve Student Satisfection ]
40                         {ReduceStudentConcern; ImproveTeachingQuality};
41         Goal ReduceStudentConcern => [  YearlyStudentComplaints < 10 ];
42         Goal ImproveTeachingQuality => [ HighQualityLecture ];
43     measures:
44         YearlyStudentComplaints = queryRaised @Year [Display BarChart];
45         HighQualityLecture = compaintsReceived @Month [Display BarChart];
46     variables:                              // Variable Definitions
47         @parameter propensityOfTeachingPreparation::Integer;
48         @export offeredModule::Module;
49         teachingPreparationInHours ::Integer;
50         lectureDelivered::Integer;
51         lectureMissed:: Integer;
52     subscribes: LectureSlot;              //Subscribe Time Event
53     produces:
54        @trace DeliverLecture(Module module, Integer hours);
55     internal-events:TeachingPreparation(Module module, Integer hours);
56     actions:
57     Action ActOnQueries:  On [(Sequence [StudentQuery(d1),StudentQuery(d2),
58                     StudentQuery(d3),StudentQuery(d4)
59     ]) And  (! StudentComplaint(s1,d5))]
60     {
61         queryResolution([d1,d2,d3,d4])
62     }
63     Action ActOnComplaint:  On [StudentComplaint(severity,description)]
64                 Where (severity > 'High') { compiantResolution(severity,description) }
65     levers:
66        Lever IncreaseTeachingPreparation : 'Lever Description' Apply [
67           propensityOfTeachingPreparation:=
68                 propensityOfTeachingPreparation + propensityOfTeachingPreparation * 0.1;
69           Ignore StudentComplaint;
70        ];
71      }
72 }
```

Figure 6.6 Illustration of an extended OrgUnit

`Parameter` should be provided while instantiating an `OrgUnit`. The exported variable can be accessed from other `OrgUnits`. A `Variable` augmented with `trace` indicator captures the value of the `Variable` at every occurrence of `TimeEvent` as trace of an `OrgUnit`.

The `TimeEvent` can be subscribed using `subscribes: <list of TimeEvents>` as shown in line 27. An `IncomingEvent` can be specified by defining its parameters, *e.g.* `<event name>(parameter list along with their types)`, as shown in line no 30 and 31. An event can be added into the traced element by augmenting `@trace` as shown in '*StudentQuery*' and '*StudentComplaint*' definitions in line 30 and 31. The `Functions` of an OrgML specification are defined using:

```
<return type> <function name>(parameter list with their types)
  { <behavioural statements> }
```

For an example, '*queryResolution*' `Function` consumes a set of queries as a list of strings and implement its behaviour to resolve those queries as shown in line 33. The inheritance relationship is illustrated using *TeachingAcademic* `OrgUnit` that inherits from *Academic* `OrgUnit` is shown in line 36.

A detailed specification of *TeachingAcademic* `OrgUnit` is shown in Figure 6.6. The specification describes goals (line 38–42), goal decomposition (line 40), measures that uses the variables of base `OrgUnit`, *i.e.*, '*queryRaised*' and '*compaintsReceived*' of *Academic* `OrgUnit` (line 44–45), a set of variable definitions (line 46–51), and a new time subscription (line 52). Specification defines *produces*, *internal-events*, *actions* blocks. The definition of `OutgoingEvents` are specified using `<event name>(parameter list along with their types)` within *produces* block. An example is shown in line 54. The `InternalEvents` are also specified in a similar manner as shown in line 55.

The scenario to specify `Actions` that contain complex event specification (line 57), state variable evaluation (line 64), function invocation (line 61 and 64), invocation of functions which are defined in a base `OrgUnit` (line 61), and parameter mapping from event to function call (line 57–61) are also illustrated in the figure. For example, `Action` '*ActOnQueries*' detects a sequence of four student queries, and in the absence of a student complaint it performs query resolution by invoking '*queryResolution*' `Function` with four query strings as a parameter list as shown in 57–59. `Action` '*ActOnComplaint*' detect a student complaint, evaluates its severity and resolves it immediately if the severity is '*High*' as shown in line 64 and 65.

An illustrative `Lever` specification is shown in line 66–70. As shown in the specification, the lever '*IncreaseTeachingPreparation* contains two lever statements – the first statement changes '*propensityOfTeachingPreparation*' by 10% and *Ignores* consumed '*StudentComplaint* event.

6.2.2 Language features

A typical *language workbench* [79] supports a common set of features to the language users for their convenience. A set of commonly seen features of domain specific languages are presented using a *feature model* in [76]. This subsection first discusses the language workbench features as described in [76] and then it highlights the subset of the features, which are implemented in OrgML workbench.

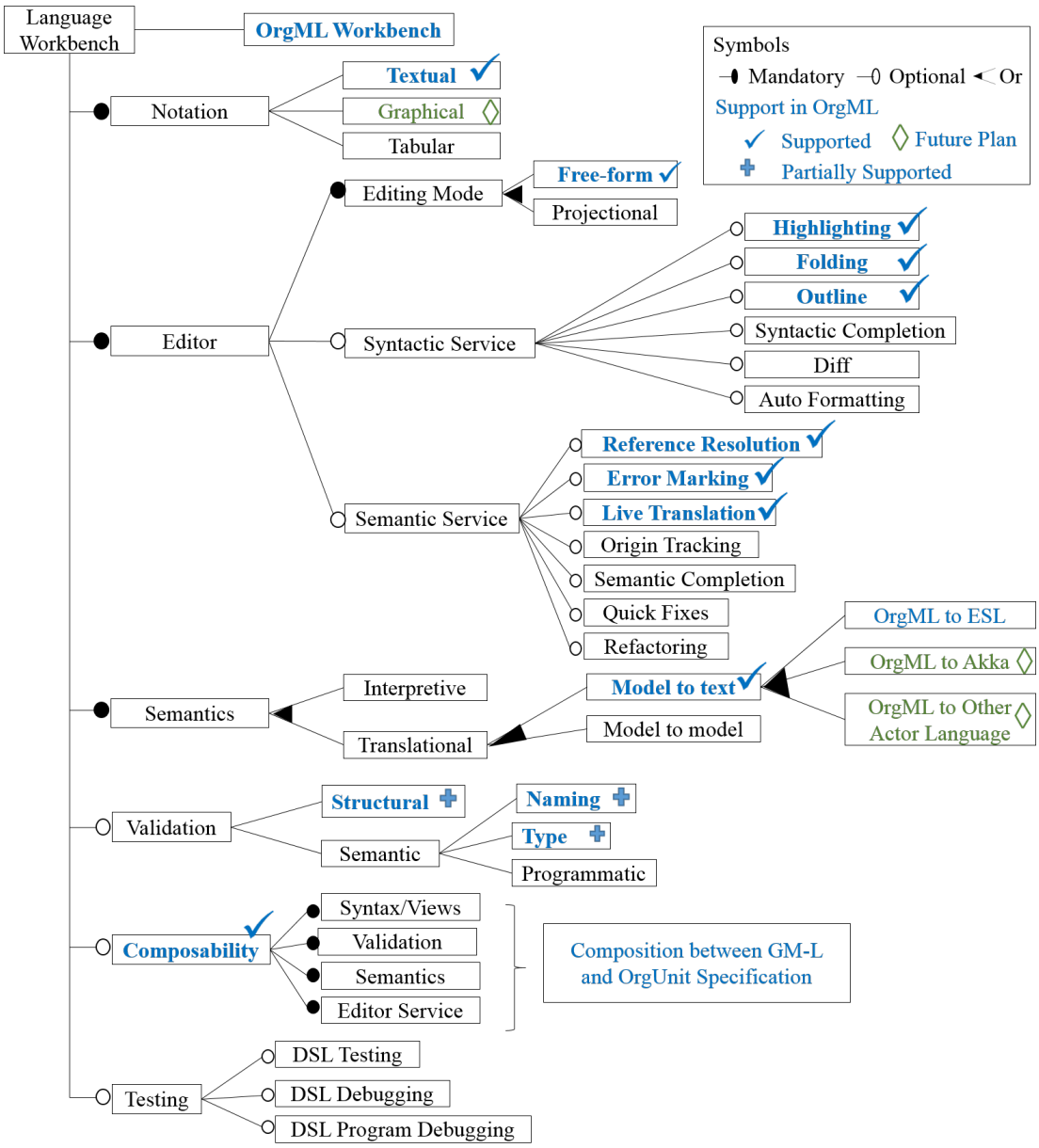


Figure 6.7 Language features of OrgML workbench (Source [76])

Language workbench feature model

Figure 6.7 shows the feature model presented in [76] to represent the state of the art of language workbenches in a structured and informative way. Pictorially, the mandatory features are connected through filled circle, optional features are connected using empty circle, and alternative features are shown using a filled edge connector. As shown in the figure, a typical language workbench is formed using three mandatory features: *notation*, *editor* and *semantics*, and three optional features: *validation*, *composability* and *testing*.

The *notation* of a language defines how a program or a model can be represented. Alternative representations are: *textual*, *graphical*, *tabular* or any of their combinations. The *semantics* or formal basis of a language is typically established using *translational semantics* or *interpretative semantics*. The *translational semantics* establishes the semantics of a language by providing a mapping to another established language. The translational semantics can be defined using model-to-text or model-to-model translation rules. The *interpretative semantics*, in contrast, defines the semantics of a language using semantic theory such as *denotational semantics* and *axiomatic semantics*. The former maps to a semantic domain and the latter sets up a logical theory for a language.

The other mandatory feature of a language workbench is an *editor*. Typically, the modes of editing includes *free-form editing*, *i.e.*, programmers freely edit the models/specification, and *projectional editing* where the programmers can edit only a projection of a model as described in [205]. In addition to the core editing capability, the language workbenches typically provide a set of syntactic and semantic editor services. The syntactic editor services include: *highlighting* (*e.g.*, syntax coloring and model highlighting), *outline* (*e.g.*, navigation via an outline view), *folding* to hide part of a model or specification, *syntactic completion* (*i.e.*, code assist through pre-defined templates), *diff* (*e.g.*, version control) and *auto formatting* (*e.g.*, restructuring, aligning, or layouting). On the other hand, the semantic editor services include:

- Reference resolution to navigate variable, function and other concept declarations
- Error marking in case of any error or warning.
- Live translation, *i.e.*, generation of targeted specification on the fly.
- Origin tracking that keeps track of source model of a transformed model.
- Semantic completion, *i.e.*, code assist using semantic information such as reference resolution.
- Quick fixes for errors.
- Refactoring that include renaming, move and other language-specific restructuring.

In addition to these mandatory features, the language workbenches may support a range of *validations* that include structural validation (*e.g.* containment and multiplicity relationships between language constructs/modelling concepts), name-space analysis and type checking.

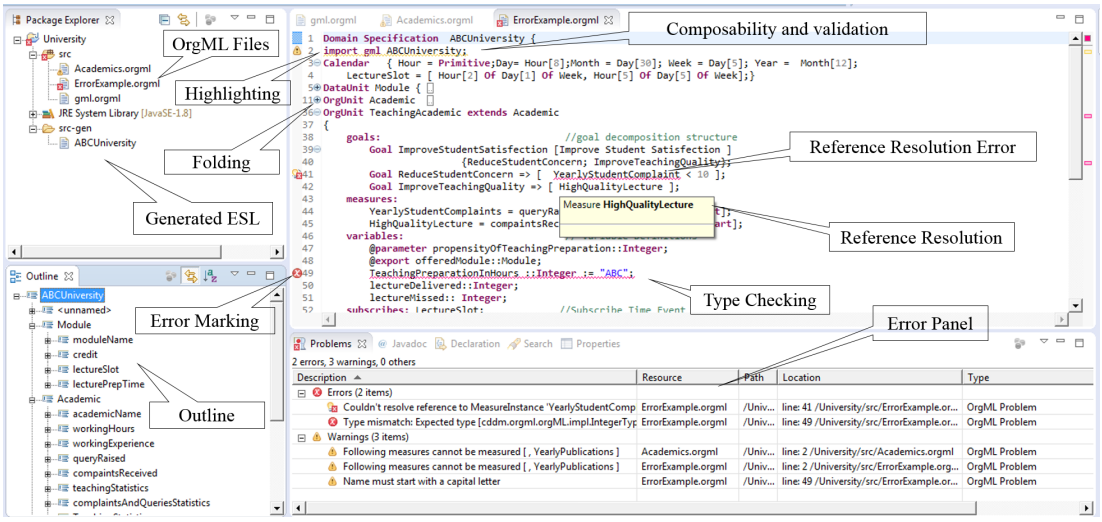


Figure 6.8 A snapshot of OrgML workbench with implemented features

The optional feature *composability* of a set of languages is a key requirement where multiple languages or specifications are needed to represent different aspects of a system (*e.g.*, GM-L Specification and Organisation Specification). The composition can be supported through *incremental extension* (*i.e.*, language integration) or *language unification* (*i.e.*, defining a unified language for a set of languages). Another optional features of a language definition is *testing* that includes: *unit testing* and *debugging* as shown in the figure.

Implemented language features

As shown in Figure 6.7, OrgML workbench supports a text-based *notation* (as shown in section 6.2), a transformational *semantics* using OrgML meta-model to ESL transformation as presented in section 5.5, a *free-form* eclipse-based *editor* with *syntax highlighting*, *folding* and *outline* features. The OrgML editors (*i.e.*, GM-L editor and OrgUnit specification editor) support semantic services that include *reference resolution*, *error marking* and *live translation* of valid OrgML model to ESL specification. In addition, it supports structural *validations*, *type checking*, and a *language unification* based language composability between GM-L specification and OrgUnit specification. A snapshot of the OrgML workbench highlighting the key features is shown in Figure 6.8. The Figure shows a folder structure of OrgML specification file, file structure of generated ESL file, Outline pane, Error description pane, and an editor canvas.

The OrgML workbench can be extended further to use the combination of graphical and textual notations. In particular, the decision making concepts of GM-L, such as `Goal`, `Measure`, `Lever`, and the structural concepts of Organisation specification, such as `OrgUnit`, `DataUnit`,

`Variable`, `Measure`, and all kinds of `Event` declarations, can be represented using graphical notations (as illustrated in Appendix B). The behavioural specification of `Actions`, `Functions` and `BehaviouralEvents` can be specified using textual specification with *projectional* editing capability. In addition, the current implementation establishes language semantics using OrgML to ESL translation rules, which can be extended by defining the transformation rules from OrgML to Akka [5] (as shown in Appendix D) or to other actor languages such as Scala [90] and Erlang [12]. These features are considered as the future work of this thesis.

6.2.3 Implementation details

OrgML workbench is implemented using Eclipse Xtext[2] language, Xtend[3] model transformation language and MWE2[4] [39] workflow engine.

Technically, Xtext is a flexible open-source framework and an expressive Java dialect to define and develop new domain specific languages. It supports an Extended Backus-Naur-Form (EBNF)-like syntax to specify language syntax and provides Java interfaces (*e.g.*, `IScopeProvider`, `IConcreteSyntaxValidator` and `ISyntaxErrorMessageProvider`) and abstract Java classes (such as `AbstractDeclarativeValidator` and `AbstractGenerator`) to implement language features that include editor, type-checker, scope, validation and language translation. Internally, Xtext uses *ANTLR*[5] as the underlying LL(k) parser technology to generate a parser for the new domain specific language. The generated parser then translates a domain specific specification into Eclipse `ECore`[6] based Abstract Syntax Tree (AST) that can be traversed, evaluated and transformed using Java-based accessors classes and APIs. The custom scoping, type-checking, structural validations, value converter and formatter can be realised by implementing the provided Java Interfaces and abstract classes. In addition, Xtext supports Xtend language to specify model-to-model and model-to-text transformation rules. The transformation rules can be specified using Xtend language (as discusses in section 5.5) and integrated with Xtext using MWE workflow engine. To realise the OrgML workbench, the following Xtext modules are implemented:

- **Parser**: The concrete syntax of OrgML grammar is specified using Xtext format by unifying the syntax of GM-L Specification and Organisation specification, which are

[2] http://www.eclipse.org/Xtext/
[3] http://www.eclipse.org/xtend
[4] help.eclipse.org/kepler/topic/org.eclipse.xtext.doc/contents/118-mwe-in-depth.html
[5] www.antlr.org
[6] https://www.eclipse.org/emf

Table 6.1 OrgML validation rules

Validation rule	**Description**	**Category**
Naming convention	OrgUnit, Goal, Measure, Lever and Events should start with a capital letter, variables must start with a lower case letter	Warning
Duplicate name	All element names of an OrgUnit and DataUnit should be unique	Error
Duplicate argument names	All parameters of an event must be unique; All event parameters of an Action specification must be unique	Error
Measure consistency	All measures of a GM-L specification should be owned by at-least one OrgUnit	Warning
Lever consistency	All Levers of a GM-L specification should be specified by at-least one OrgUnit	Warning

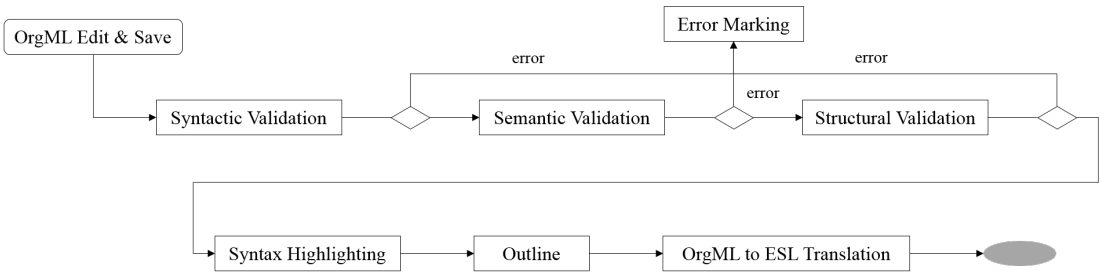


Figure 6.9 Implemented OrgML workbench workflow

presented in Figure 6.3 and Figure 6.4. All left-recursions of the syntax presented in Figure 6.3, Figure 6.4 and Figure 5.14 are removed by left-factoring the grammar[7].

- **Scoping**: The customised scoping rules are implemented for inherited `OrgUnit` elements, event parameters, functions parameters, and action parameters by implementing `IScopeProvider`. In additional, the custom scoping rules are implemented to make `Goals`. `Measures` and `Levers` accessible across Organisation specification.

- **Validation**: The naming conventions, type-checking and structural validations are implemented by extending '`AbstractDeclarativeValidator`'. Implemented validations rules along with the error classification are presented in Table 6.1

- **Translator**: The OrgML to ESL translation rules (as defined in section 5.5 of Chapter 5) are encoded using Xtend language in a Java class class that extends abstract class `AbstractGenerator`.

[7]Left-factoring is grammar rewrite in such a way that all recursive production rules consume at least one token or character before going into the recursion

```
/* Legend:   OrgML Elements   */

An OrgML program P := <Org, cal>, where
 Org   : A set orgs where each org represents an OrgUnit
 cal   : Calendar specification that defines a set of TimeEvents.

Definition of org:
  org := <init_0, state, trace, inbox, Time, Inp_E, Internal_E, Out_E, Act, Behav>, where
    init_0    : Initial state of an OrgUnit. Initial state is defined using Parameter values
     .
    state     : State of an OrgUnit. State is specified using OrgUnit Variables.
    trace     : Event trace of an OrgUnit.
    inbox     : Event queue of an OrgUnit. It contains a subset of {Time,Inp_E, Internal_E}.
    Time      : A set of subscribed TimeEvents.
    Inp_E     : A set of InputEvents.
    Internal_E: A set of InternalEvents.
    Out_E     : A set of OutgoingEvents.
    Measure   : Set of Measures.
    Act       : Set of Actions.
    Behav     : Behavioural units of Action, BehaviouralEvent and Function, where
                    All behav ∈ Behav =  {stmt_1, stmt_2, stmt_k} where stmt_i is a behavioural
    statement

Definition of cal:
  cal := {time_1, time_2, time_m},  where time_i ∈ TimeEvent

Definition of Levers:
  L := {lever_1, lever_2, lever_n},  where lever_i is a Lever specification

Algorithm:
A simulation is the execution of an OrgML program P = <Org, cal> with a lever_p for x
    iterations of time_s where lever_p  = ∈ {∅, L}, and time_s ∈ cal. In an OrgML program
    execution, all orgs execute in parallel where each org takes out events from its inbox
    , update trace information, evaluates trace and state conditions and performs
    behaviours, which are true at a given moment as follows:

execute(OrgML P, L lever_p, Time time_s) {
 P= transform(P,lever_p)
 occurrence_ts := 0
 while (occurrence_ts  < x) {
   compute non-primitive time of cal
   ∀  t_i ∈ cal {
     if (t_i = true)
      send t_i to all orgs that subscribe t_i
     if (t_i = time_s ) { occurrence_ts := occurrence_ts + 1}
   }
   ∀ org ∈ Org {
     ∀ event_i ∈ inbox {
       trace := <trace,event_i>                              //Update trace
       execute (behav of event_i , org) , where behav ∈ Behav of org
     }
   }
 }
}
transform(OrgML P, L lever) {
 ∀ org ∈ Org {
  if (lever is applicable for org) {
    transform org specification by considering lever definition.
  }
 }
}
execute(Behaviour behav, Org org) {
  ∀ stmt_k ∈ behav {                                    //For all statements of behaviour
     behav
   case (stmt_k)
   {
     assignment ⇒ { <init_0 | state > →  state_new  }       //Assignment Statement; Update
    State
     new        ⇒ { Org := {Org, new org}                  //New of OrgUnit; Update OrgUnit
    set
     send(event) ⇒ { trace := <trace,event>; send event } //Send Statement; Update Trace
    and send Event
   }
   traceBasedExecution(org)
  }
 }
}
traceBasedExecution(Org org) {
  ∀ action_i ∈ Act of org {
   ec = Evaluate event condition of action_i with respect to trace of org
   es = Evaluate state condition of a_i with respect to state of org
    if (ec && es) { execute(behav of action_i, org) }
  }
  ∀ measure_j ∈ Measure of org {
   if (event condition of measure_j match trace) { display measure_j }
  }
 }
```

Figure 6.10 Execution of OrgML specification

Finally, a workflow is configured using MWE2[8] specification to synchronise workbench related activities as shown in Figure 6.9. The edit and save of an OrgML specification (specified in a file with file extension'`orgml`') triggers validation rules that include'*Syntactic Validation*', '*Semantic Validation*' and '*Structural Validation*'. The detection of an error in any of the validation step terminates the flow and triggers '*Error Marking*' activity. The flow without any error condition triggers '*Syntax Highlighting*' action. Subsequently the workflow updates '*Outline*' and invokes transformation rules (*i.e.*, OrgML to ESL transformation module).

6.2.4 Execution of OrgML specification

A high-level execution schema of an OrgML specification is shown in Figure 6.10. An OrgML specification that contains a set of `OrgUnits` and a `Calendar` is simulated for `TimeEvent` $time_s$ with or without a `Lever` $lever_p$ to understand the as-is behaviour of an organisation or the consequence of `Lever` $lever_p$ over time unit $time_s$. To execute an OrgML specification, the specification P is first transformed into a new specification by applying `Lever` specification $lever_p$. The `Calendar` and `OrgUnits` are then executed in parallel.

Semantically, the `Calendar` evaluates non-primitive `TimeEvents` and sends `TimeEvents` to all subscribed `OrgUnits`. All `OrgUnits` concurrently processes events, which includes `TimeEvents`, `IncomingEvents` and `InternalEvents`, from their respective *inbox* or event queue. Each `OrgUnit` takes out an event from its *inbox*, updates trace information by appending event to its trace information, assesses the applicability of `Actions` by evaluating the event trace and state condition of `Action` specifications, and performs behavioural specification for all valid `Actions`. While performing a behavioural specification, an `OrgUnit` perform a sequence of `Statements` that may change its state, can trigger `InternalEvents`, may send `OutgoingEvents` to other `OrgUnits`, and may produce new `OrgUnits`.

The execution semantics is realised by translating `Calendar` and `OrgUnits` into ESL *actors* by applying the transformation rules described in section 5.5.

6.3 OrgViz Data Visualiser

OrgViz Data Visualiser is a customisable and extensible graphical display unit that visualises simulation results using user-specified form. It visualises OrgML `Data` that includes the

[8] help.eclipse.org/kepler/topic/org.eclipse.xtext.doc/contents/118-mwe-in-depth.html

numerical values of `Measures`, `Traces` and information about the occurrences of `Event` (*i.e.*, OrgML `EInfo`).

The concept of *temporal data model* presented by Goralwall et al. in [83] and their visualisation techniques discussed in [179] are considered for realising the OrgViz Data Visualiser. Principally, a *linear* order 2D display unit that conforms to *relative* temporal structure over *discrete* time domain [179] is conceptualised and implemented to visualise OrgML `Data`. This section presents an overview of the temporal data model, correlates the temporal data model with OrgML concepts and presents the design considerations and implementation details of OrgViz Data Visualiser.

6.3.1 Temporal data model and visualisation

A typical temporal data model [83, 179] is characterised along four dimensions: *temporal structure*, *temporal domain*, *temporal order* and *temporal history*. Temporal structure introduces a time domain using two types of time definitions: *temporal primitives* and *derived definitions*. A temporal primitive indicates a single time instance. It can be of two types: *absolute* time or *relative* time. An example of an absolute time is `January 1, 2018`, whereas the examples of relative time are: `first day of an academic session`, `first hour of a working day`, *etc*. A derived time definition is defined using *time interval* and *time span*. `Every first week of month` and `from January to July` are the examples of time interval and time span respectively.

The *temporal domain* is classified into two broad categories: *discrete* and *continuous*. The discrete time domain is isomorphic to natural numbers, whereas the continuous domain is isomorphic to real numbers. The *temporal order* can be *linear* or *branching*. A linear order time is represented using an unidirectional time axis. In contrast, the future is not determined and the time dimension divides into multiple paths for branching order. The *temporal history* is a sequence of related temporal entities where each temporal entity contains a tuple `<data, time>`. The *data* represents the numerical value and *time* indicates a time information. The temporal history can be classified into three types based on the type of time information. The types are: *valid*, *transaction* and *event*. In *valid history*, the data is augmented with a *absolute* time information, *transaction history* uses transaction time (such as database entry) as time information, and *event history* considers instantaneous facts or events to represent time information.

The visualisation of temporal data models is a well studied research area [4]. It focuses on three aspects *time*, *data* and *representation* [4] where *time* is a quantifiable information that can be mapped to an axis, *data* is a set of values that need to be analysed, and *representation* is a format to display the data in a meaningful manner. The representation is largely dependent on how the data is tied up with the time. For example, the *Line* graph is frequently used to represent variables that change over time, whereas the changes of multiple variables can be effectively represented using *Bar* chart, *Stacked Bar* chart, *etc*. The key factors to represent a temporal data model (as presented in [4]) are discussed below:

- Time point vs. time interval: valid *data* at a (absolute or relative) time point can be represented without concerning about time information, whereas time-interval and time-span expect the time information to be represented in a visualisation.
- Linear vs. branching: the time information of a linear time data can be represented using a single axis whereas the branching time requires complex graph structure to represent time dimension.
- Univariate vs. multivariate: an univariate data (*i.e.*, single data value) can be represented as they exist whereas visualisation of multivariate data (*i.e.*, multiple data values) needs data preprocessing, composition and consolidation.
- Data vs. data abstraction: Visualisation of valid data may not be possible for all scenarios. Data abstraction techniques, such as data aggregation, summation, and means, are useful for such scenario.
- Dimensionality (2D vs. 3D): This characteristic simply distinguishes between 2D and 3D representation. The 2D data visualisation is mostly used in numerical data representation.

An OrgML data visualisation specification describes: what `Data` needs to be captured and visualised (*i.e.* selection of `Variables`, `Traces` and `Measures`), when they should captured/visualised (*i.e.*, based on which `TimeEvent` and/or other `Events`) and how they should be represented (*e.g.*, ChartType). In addition, the specification captures additional information about visualisation characteristics in the form of a meta-data. The meta-data captures information such as: caption of a graph and *data abstraction* requirement (*e.g.* data aggregation, summation, and means). Therefore, a `Data` visualisation specification describes *data*, *time* and (*user-defined*) *representation* as suggested in [4].

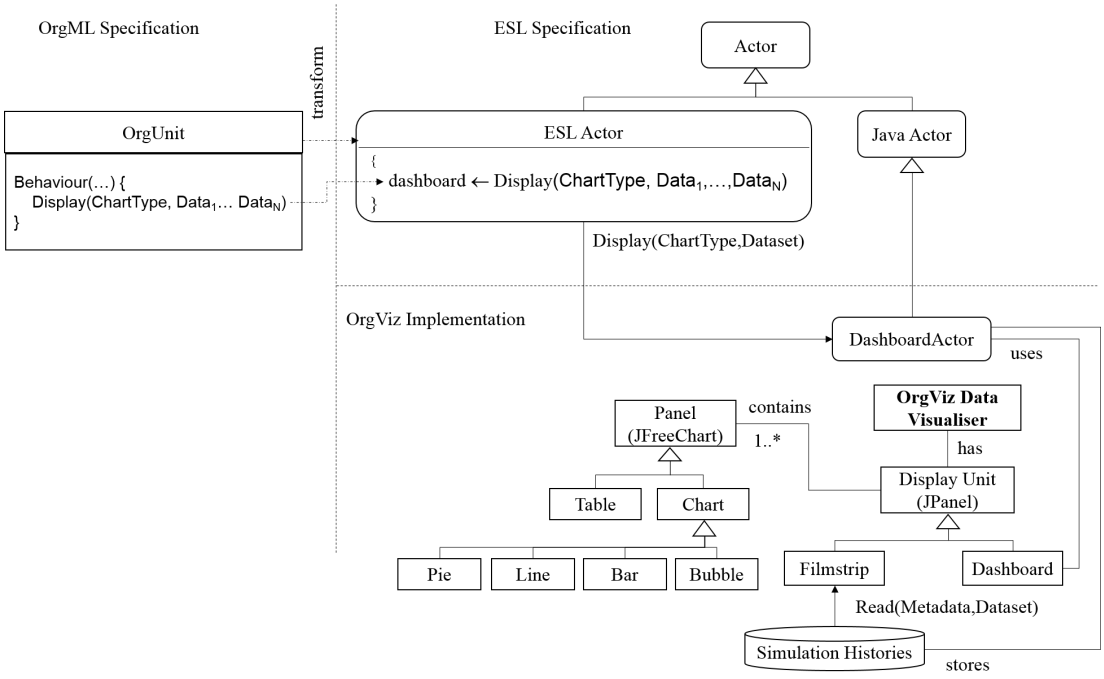


Figure 6.11 Implementation details of OrgViz Data Visualiser

Integration of temporal features with OrgML concepts define the scope and design considerations of OrgViz Data Visualiser. The OrgML `Calendar` definition and set of `Event` specifications introduce a temporal structure. In particular, they define a set of relative temporal primitives, time-interval and time-span. The concept of periodic primitive `TimeEvent` and discrete nature of OrgML `Event` definitions limit the visualisation scope to discrete time domain. The numerical `Values` of OrgML `Data` defines the representation scope to 2D linear order graphical representations. Moreover, the characteristics of `Measures` and `Traces` closely relate to event history where they expect 2D linear order graphical representations. The comparative analysis of multiple `Measures` expects multivariate visualisation (in addition to *univariate* visualisation).

The next section presents implementation details of OrgViz Data Visualiser that realises the aforementioned characteristics.

6.3.2 Implementation details of OrgViz Data Visualiser

The OrgViz Data Visualiser is a 2D linear order *Display Unit* that visualises the numerical values of OrgML `Data` in a user-defined form. It supports two display modes: *Dashboard* and *Filmstrip* where the Dashboard is an active display unit that synchronises with discrete

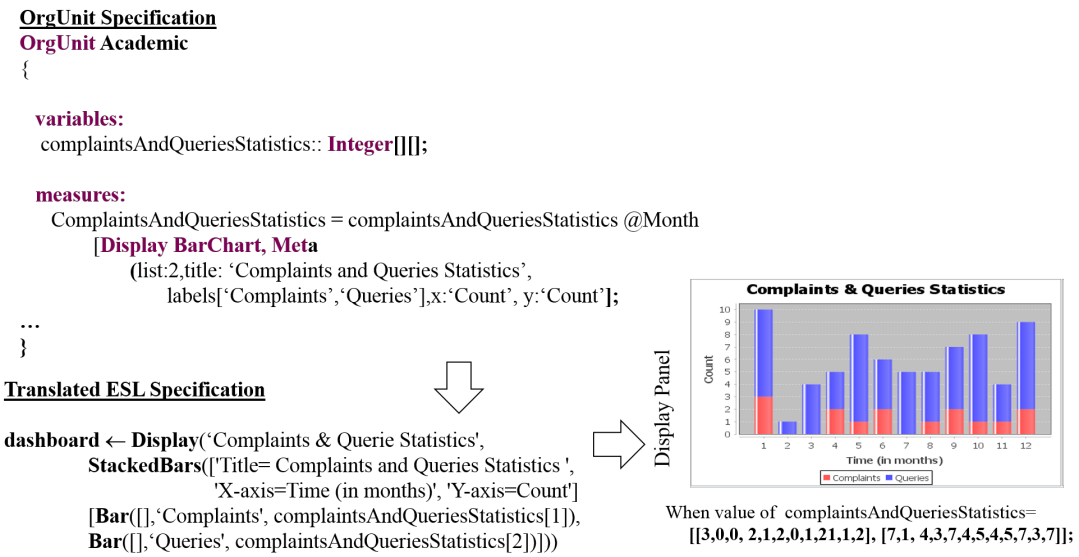


Figure 6.12 Code fragments for OrgViz Data Visualiser

`Events` of a simulation. Filmstrip, in contrast, is a repository-centric slider-driven interactive user interface that visualises and navigates simulation data of historical simulation runs. It visualises a linear event history using 2D graphical representation.

The implementation details of the OrgViz Data Visualiser that includes Dashboard and Filmstrip is shown Figure 6.11. As shown in the figure, both, Dashboard and Filmstrip, inherit from a common *Display Unit*. The display unit is a container that contains multiple *Panels*. Each *Panel* can display either a table or a chart to show numerical values of one or multiple `Variables`, `Traces` and `EInfo` (*i.e.*, univariate and multivariate display). The *Display Unit* unit is capable of computing data composition, consolidation and other data abstraction techniques (*e.g*, `sum`, `average`, `means`) to support multivariate data visualisation and data abstraction.

Technically, Display Unit is a Java JPanel[9]. It supports a configurable multi-tab multi-panel layout to visalise one or multiple `Variables`, `Traces` and/or `EInfo`. The Panels are realised using Jfreechart[10] based visualisations, such as Line chart, Bar chart and Pie chart. The specialised behaviour of Dashboard and Filmstrip are described below:

1. Dashboard: Dashboard concurrently operates with the ESL simulation engine, interacts with ESL actors to collect relevant values at specific `Events`, and displays simulation data using user-defined graphical form.

[9] https://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html
[10] www.jfree.org/jfreechart

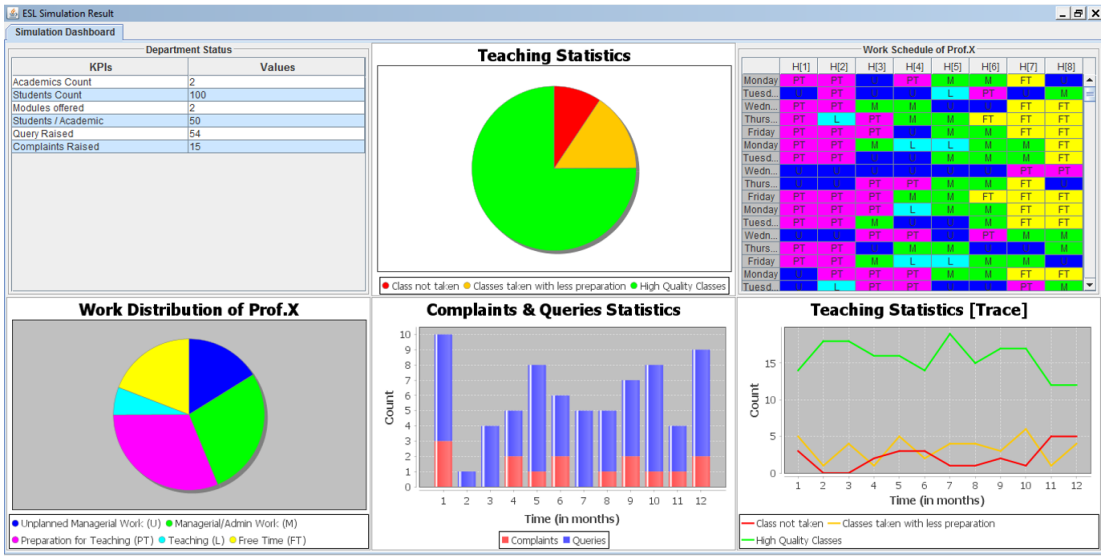


Figure 6.13 An illustration of Dashboard

Dashboard uses a specialised Java actor, termed as *DashboardActor*, which runs concurrently with ESL actors to collect information from ESL actors. Each ESL '`Display`' statement triggers a '*Display*' message from ESL actor to DashboardActor actor. DashboardActor delegates '`Display`' message to Display Unit for preprocessing (if required for data consolidation and data abstraction) and data visualisation. In addition, DashboardActor stores message dataset along with chart-type, meta-data to *Simulation Histories* repository for future use.

Operationally, each simulation run instantiates a singleton DashboardActor instance with a static reference, termed as '`dashboard`', and all ESL actors use the static reference to delegate their '`Display`' messages to DashboardActor as shown in Figure 6.11. The OrgML to ESL translation rules delegate '`Display`' messages from ESL actor to DashboardActor.

A sample OrgML specification with '*ComplaintsAndQueriesStatistics*' `Measure`, translated ESL code and visualisation of '*ComplaintsAndQueriesStatistics*' `Measure` are shown in Figure 6.12.

2. Filmstrip: Filmstrip fetches historical simulation data from Simulation Histories repository and uses functionalities of Display Unit to visualise and navigate stored simulation data as shown in Figure 6.11. It provides a slider to navigate the time axis of a simulation run.

A snapshot of a Dashboard is shown in Figure 6.13 as an illustration. The Dashboard visalises a set of `Measures` and `Traces` of a *Department* of ABC University using six *Panels*.

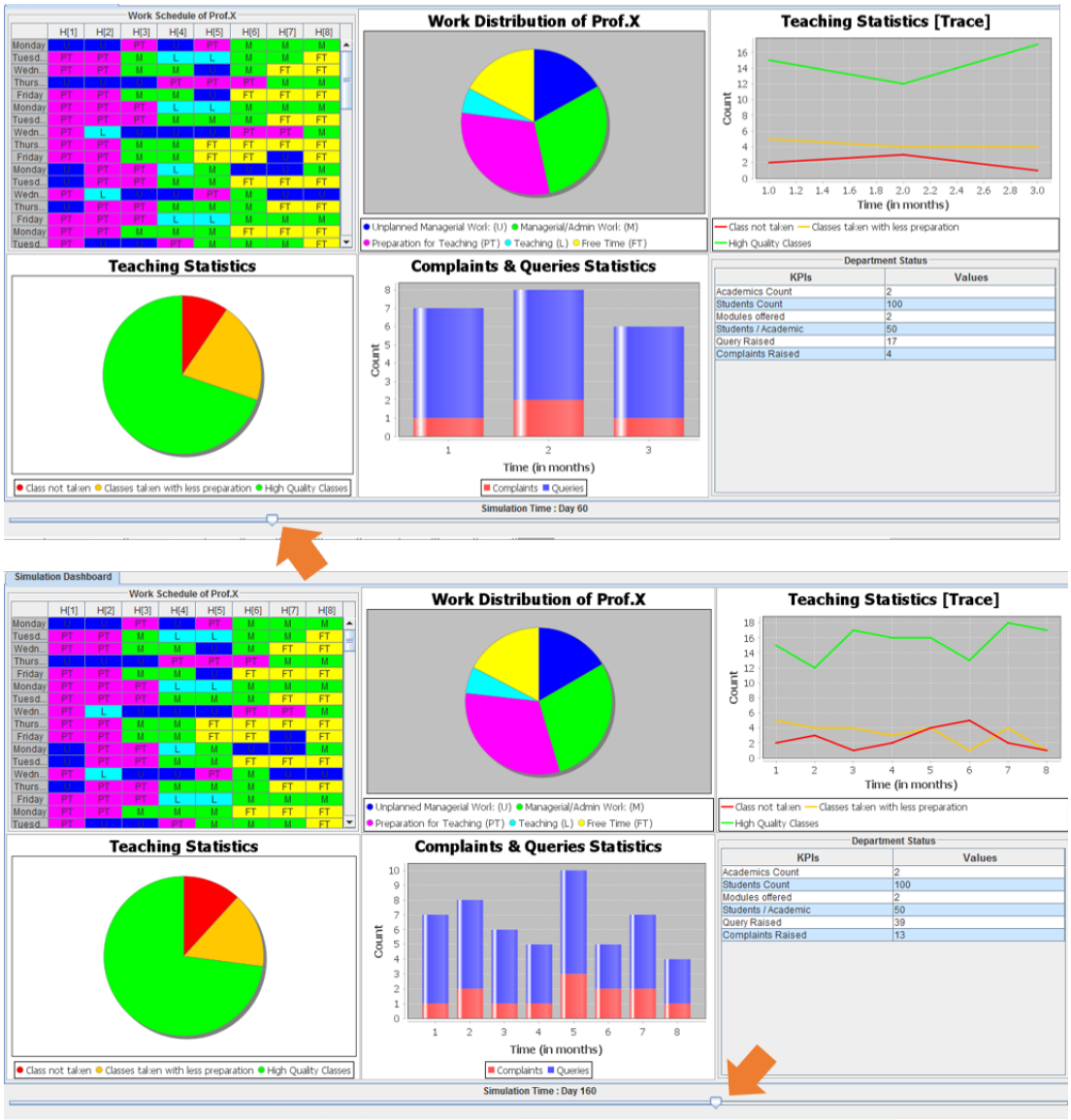
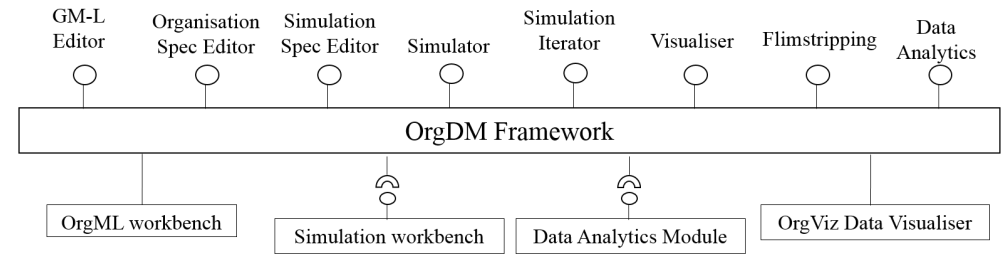
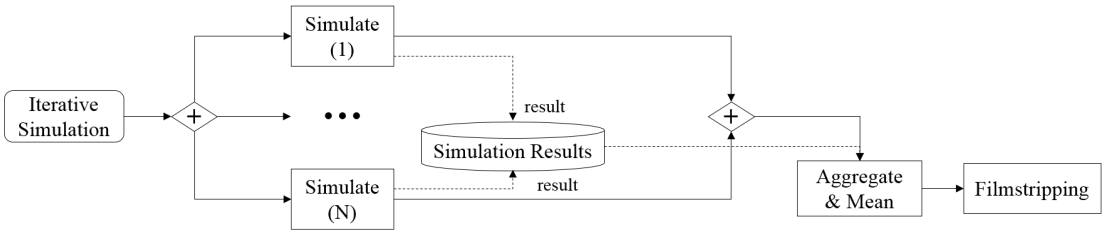


Figure 6.14 An illustration of Filmstrip

Panel '*Department Status*' shows the key `Measures` of the Department (such as number of academics, number of students, number of modules offered by the department) in a tabular form, panel '*Teaching Statistics*' shows a consolidated view of lecture status (*i.e.*, how many lectures are not taken by the academics, how many lectures are delivered with less preparation, *etc.*) using a pie chart, panel '*Work Schedule of Prof.X*' shows work schedules of an academic (*i.e.*, an *event history* of work schedules) using a tabular form, panel '*Work Distribution of Prof.X*' shows the work distribution of an academic, panel '*Complaint and Query Statistics*' shows the histories of complaints and queries, and panel '*Teaching Statistics [Trace]*' shows the histories of teaching statistics (trace of '*Teaching Statistics*'). The Dashboard is an active display unit



(a) OrgDM Plugin Architecture



(b) Simulation Iterator Workflow

Figure 6.15 OrgDM capabilities and workflows

that updates panel graphs as the simulation progresses. Whereas, the Filmstrip provides a slider to navigate time axis as shown in Figure 6.14 where two different points in time are highlighted using arrow for illustration.

6.4 A decision making framework

A configurable and extensible actor-based simulation framework, OrgDM, is comceptualised to support organisational decision-making using the proposed technology aids. An OrgDM framework supports eight core capabilities in an integrated manner as depicted in Figure 6.15 (a). The capabilities are: (i) GM-L editing, (ii) Organisation specification editing, (iii) Simulation specification editing, (iv) Simulation (v) Iterative Simulation, (vi) Visualisation, (vii) Filmstripping and (viii) Data analytics.

An OrgDM framework uses OrgML workbench and OrgViz Data Visualiser. It supports two extension points: *simulation workbench* and *data analytics*. The *simulation workbench* is designed to plug-in a simulation language editor and a simulation engine. The ESL workbench [54] provides the *simulation workbench* extension point in this research. The data analytics extension point provides advanced data analysis on simulation results.

In this formation, OrgML workbench supports GM-L editing and Organisation specification editing. OrgViz Data Visualiser supports visualisation and filmstripping capabilities. ESL

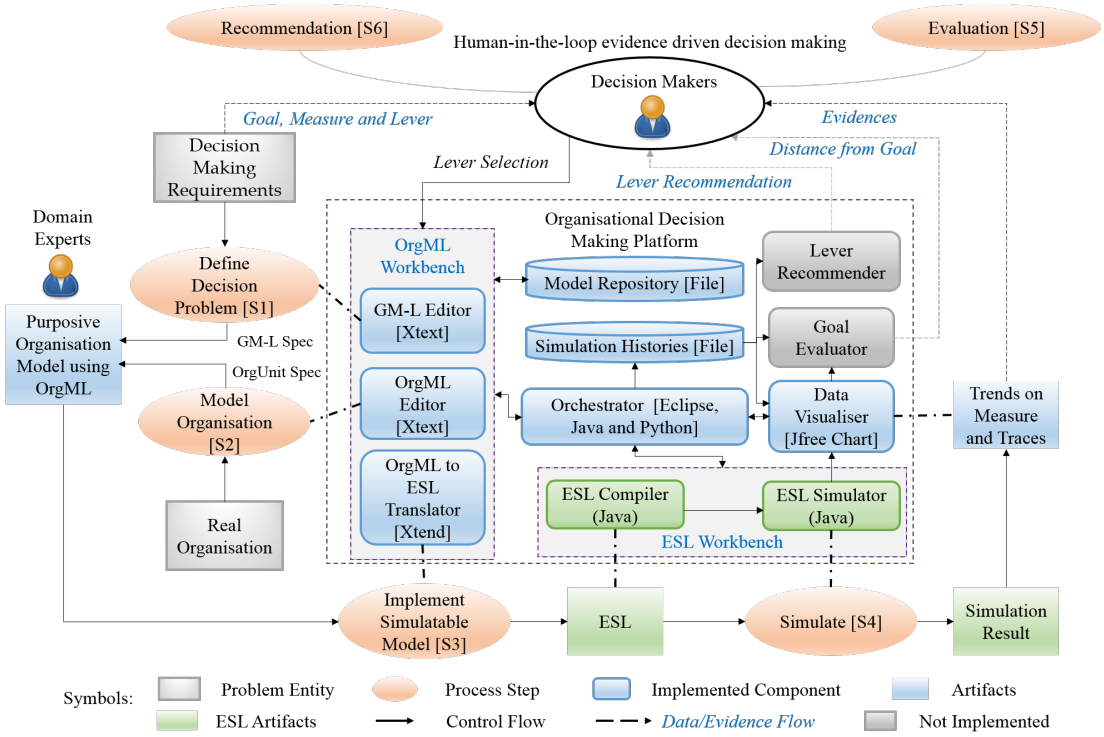


Figure 6.16 Architecture of organisational decision-making framework

workbench supports *Simulation specification editing* (*i.e.*, ESL editing) and *Simulation*. In addition, OrgDM framework implements a workflow that realises *Iterative Simulation*. The action iterative simulation (concurrently) simulates same specification multiple times and shows the consolidated data as depicted in Figure 6.15 (b). The support for iterative simulation establishes the statistical significance of simulation results[11].

The proposed OrgDM framework is realised using a Java and Python based toolset on Eclipse platform. The rest of this section discusses a high-level tool architecture of the proposed OrgDM framework and the realisation of the proposed decision-making method using OrgDM framework.

6.4.1 Tool architecture

A tool architecture of OrgDM framework is illustrated in Figure 6.16. As shown in the figure, OrgDM framework contains an orchestrator, two file-based repositories (*i.e.*, *model repository* and *simulation histories*) and four functional building-blocks: (i) OrgML workbench, (ii) ESL workbench, (iii) OrgViz Data Visualiser and (iv) a data analytics module. The orchestrator is a

[11]This is required as the organisation model contains several uncertainties and probabilistic behaviours. Moreover, the overall macro-behaviour of an organisation emerges from multiple micro-behaviour.

broker that establishes the interoperability among the plugged-in building blocks, communicates with repositories, implements *Iterative Simulation* workflow and enables expected data- and control-flow across components as shown in the figure. The orchestrator is realised using the Eclipse plugin architecture[12], a set of Java libraries to store and retrieve data to/from repositories, and a Python[13] module for data consolidation and data abstraction. The Python modules use Python Pandas[14] – an open source easy-to-use data structures and data analysis tools for the Python programming. However, it can be extended with Python SciPy[15] (a Python library for scientific computing) to support advanced data abstractions.

The OrgDM framework uses Eclipse, Xtext and Xtend-based OrgML workbench implementation (as described in section 6.2), ESL technology as introduced in section 5.2.2 of Chapter 5, and Java Jfreechart based OrgViz Data Visualiser (as presented in section 6.3). It considers two file based repositories. The *Model Repository* contains the OrgML specifications as an Eclipse project and *Simulation Histories* stores simulation results using JSON format[16] in a dedicated directory folder.

In addition, two advanced data analytics modules, *Goal Evaluator* and *Lever Recommender*, are conceptualised (but not implemented) to interpret stored simulation results and provide advanced insights to the decision-makers. The key objectives and possible implementation techniques of these modules are discussed below:

- **Goal Evaluator**: The primary objective is to show the distance of a simulation result from the desired goals.

- **Lever Recommender**: The key objectives is to help decision-makers along three dimensions – (i) identify most sensitive `Lever`(s) so that they can be explored/refined further, (ii) recommend variants of existing *Levers* that may have better potential to achieve desired organisational `Goals` and (iii) recommend a large numbers of random `Levers` to avoid being trapped in a local optimum.

[12] https://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html
[13] https://www.python.org
[14] https://pandas.pydata.org
[15] https://www.scipy.org
[16] https://www.w3schools.com/js/js_json_intro.asp

6.4.2 Method realisation

Figure 6.16 highlights how the OrgDM framework helps to perform the organisational decision-making method steps proposed in Chapter 5. As shown in the figure, GM-L editor helps to capture decision problem in method step S1, the OrgML editor assists domain experts to capture organisational aspects in S2, the OrgML-to-ESL translation rules helps to implement simulation model in S3 (an automated transformation of OrgML to ESL is also conceptualised), and the ESL simulator and the OrgViz Data Visualiser collectively help decision makers and domain expert to simulate and observe simulation results in S4 step. In the current implementation, decision makers evaluate simulation results (*i.e.*, method step S5) and recommend levers (*i.e.*, method step S6) based on observations of multiple simulations. The proposed Goal Evaluator will compute and show the distance of observed `Measures` from the desired `Goals` and Lever Recommender will help to navigate the decision space by recommending new set of `Levers`.

6.4.3 Summary

This chapter proposes a set of technology aids to support the organisational decision making approach presented in Chapter 5. The key contributions of this chapter are three-fold – (i) OrgML workbench that supports standard editing capabilities to author OrgML specifications and OrgML-to-ESL translation to use ESL as a simulation engine, (ii) OrgViz Data Visualiser that adopts established visualisation capabilities to display the simulation result using user-defined form, and (iii) OrgDM framework that integrates the capabilities of OrgML workbench, ESL technology and OrgViz data visualisation to provide a platform where decision makers can specify their decision problems, domain expert can capture their organisational knowledge using organisation model, and they can observe simulation results using their specified format.

Chapter 7

Research Validation

The value of an aid for organisational decision-making comes from its practical utility and convenience to address large complex problems which are difficult to solve otherwise. The goal of this chapter is to establish the *usefulness* of research hypotheses, demonstrate the *efficacy* and *utility* of the research contributions, and applicability of the proposed approach and technology aid, which are introduced in Chapter 3, 5 and 6. An *Artificial* and *Ex-Post* validation strategy, as discussed in Chapter 2, is adopted to evaluate these research artifacts. Four synthetic yet close to real life case studies with different characteristics are modelled using OrgML and the necessary *what-if* analyses are performed (by translating OrgML specification into ESL) to produce sufficient evidence for organisation decision-making. The case studies are analysed to report the technological advances from the state-of-the-art technologies for organisational decision-making and applicability of the proposed approach in a range of decision-making problems. The limitations of the research artifacts and further improvements as the future work on this research are ascertained from the critical evaluation. With respect to the research methodology described in Chapter 2, this chapter presents an overview of *Evaluate research outcome* research activity, describes the outcome of *Demonstration and communication* research activity and focuses on *Establishing rigour* research activity.

The characteristics of the four case studies are discussed in Table 7.1. This chapter discusses the overview of three case studies. Section 7.1 discusses a case study from a Software Service Consulting domain where organisations target precise year-on-year profit margins by offering software development services in a moderately mechanistic environment. A decision-making problem for a complex, dynamic and uncertain environment is illustrated using Indian Demon-

Table 7.1 Characteristics of validation case studies

Case Study	Characteristics
Software Service Provisioning Organisation	Hierarchical and vertical organisation structure (SSPO organisation), mechanistic behaviour with minimal probabilistic distributions, stable environment (*i.e.* demand and supply can be specified using mathematical formulae)
Business Process Outsourcing Organisation	Set of competing organisations (*e.g.*, competitors), organisation definitions are monolithic, environments for all competing organisations are nonlinear and uncertain
Demonetisation	An example of emergent behaviour (definition of *Society*), autonomous and adaptive units (*e.g*, *Citizens*), significant uncertainty and nonlinearity
University	Hierarchical and vertical organisation structure (*e.g.* University and Department), uncertain behaviours (*e.g.*, Academics and Students), emergent behaviour (*e.g.* University and Department)

etisation initiative[1] in section 7.2. A decision-making scenario from ABC University (used as the running example in this thesis) is described in section 7.3. A decision problem from competitive Business Process Outsourcing (BPO) domain where multiple organisations compete with each other to maximise their profits and market shares is presented in Appendix E.

The rest of the sections of this chapter focus on critical evaluation of the research artifacts. Section 7.4 evaluates research artifacts by comparing them to the state-of-the-art technology aids, highlights technological improvements and discusses the usage and applicability of the proposed approach. The limitations, threat to validity and further improvements of the proposed approach are discussed in section 7.5. The first three sections and Appendix E collectively focus on *Evaluate research outcome* research activity. The outcome of the *Demonstration and communication* research activity is presented in Section 7.4. Sections 7.4 and 7.5 illustrate *Establishing rigour* research activity.

7.1 Software Service Provisioning Organisation

This section presents a case study that focuses on a Software Service Provisioning Organisation (SSPO) that is aiming to secure a leadership position in software service consulting space by improving its customer satisfaction, business volume and profit margin. The exploration of decision alternatives is carried out by constructing an OrgML model of SSPO, translating constructed OrgML specification into ESL, simulating ESL specification and observing simulation results using OrgViz Data Visualiser. The problem entity of SSPO, the key elements of OrgML model that captures SSPO and simulation based decision space exploration are discussed in this section.

[1] https://en.wikipedia.org/wiki/2016_Indian_banknote_demonetisation

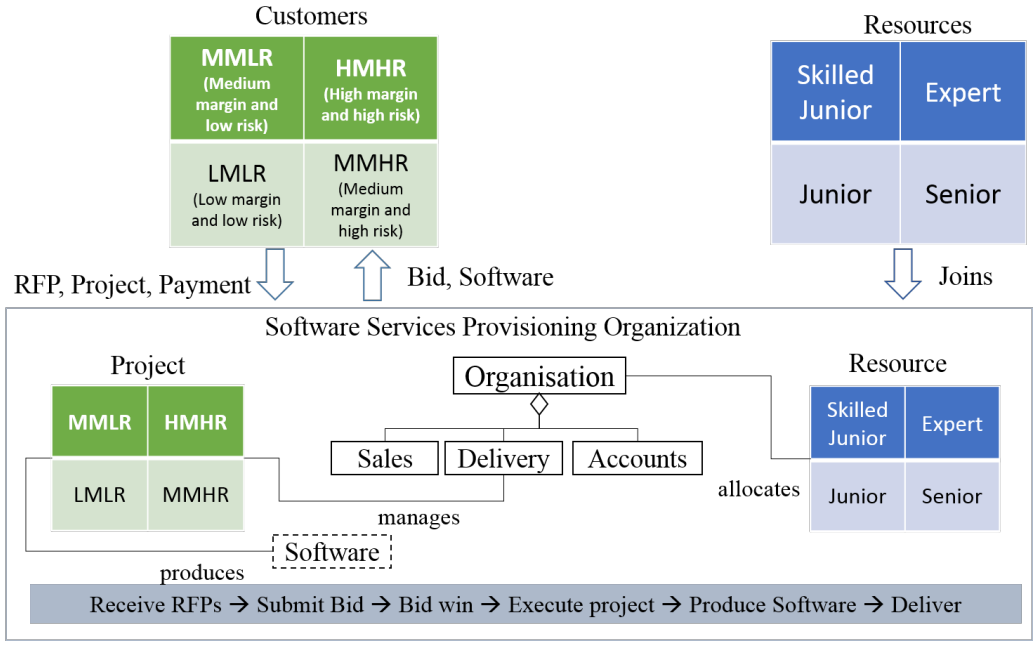


Figure 7.1 A pictorial representation of Software Service Provisioning Organisation (SSPO)

7.1.1 Problem entity

Consider a SSPO that takes up software development projects for its customers. The organisation bids for projects in response to Request for Proposals (RFPs) and wins based on its track record, projected delivery time and price considerations. It then proceeds to staff the project appropriately, executes the project using its tried-and-tested processes leading to successful delivery, winds up the completed project and then releases resources to the free resource pool. In managing this business as usual (BAU) operation, the organisation has to deal with operational complexities such as maintaining the right number of people with the right skills on its payroll, keeping enough workforce in reserve to handle incoming project demand, dealing with the attrition, ensuring maximum utilisation of existing workforce, and accounting for various delays such as hiring delay, training delay, assimilation delay *etc.*, while ensuring its business targets are met.

In order to achieve its goal of consolidating its position as leading software service provider in the market, the software service provisioning organisation needs to decide upon strategies to improve its BAU state. One strategy is simply bidding for a higher number of projects. This would result in skill improvement of the workforce over time, leading to increased productivity and quality, and thereby a good track record and improved chances of winning future bids. An additional strategy for improving bid winning percentage could be to reduce project cost

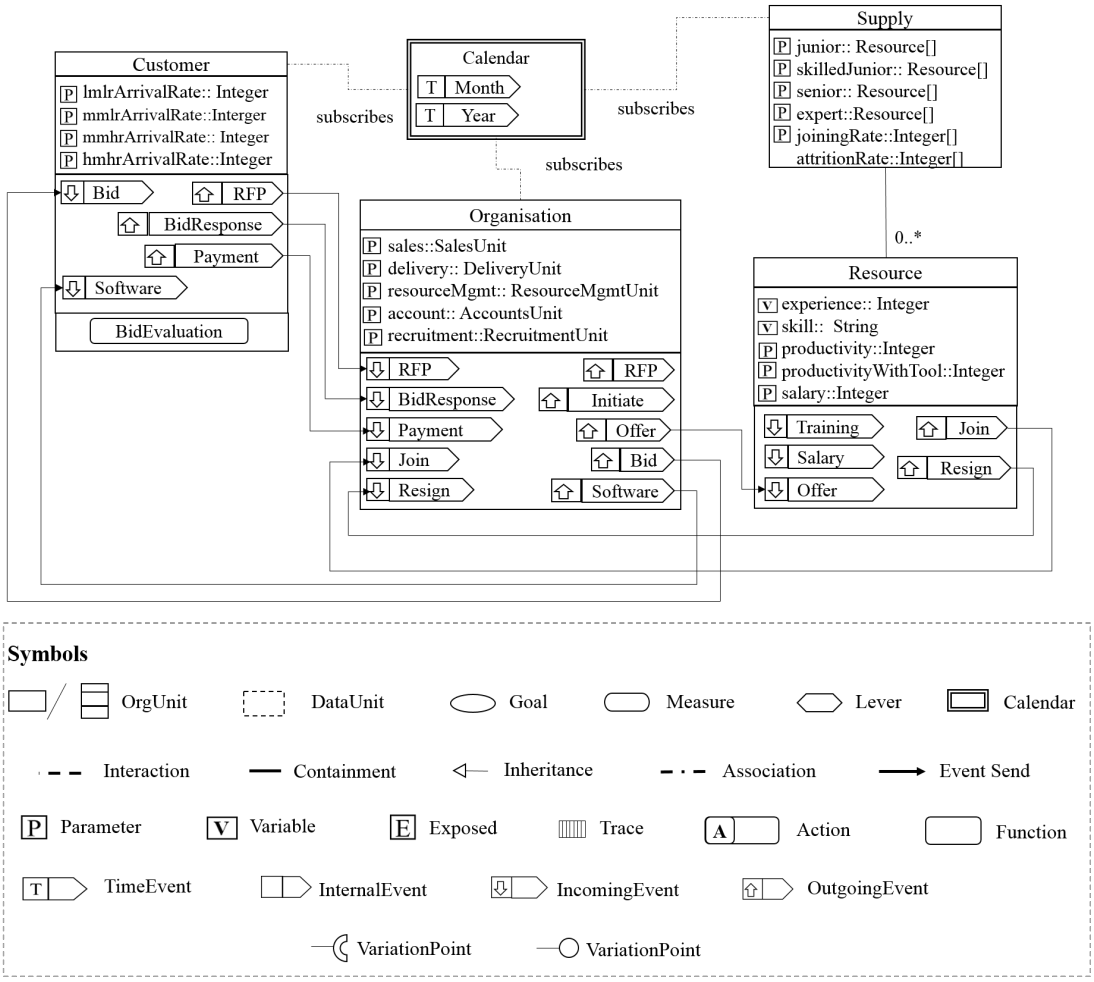


Figure 7.2 OrgML specification of Software Service Provisioning Organisation

or delivery time or both. Increased bid wins might stretch the existing workforce to the limit beyond which projects would get queued thus observing delay. A better trained workforce or use of productivity tools or both are possible strategies for managing delays but both come at some cost. Decision makers would like to know which strategy would be beneficial amongst the various alternatives.

Figure 7.1 shows an operating environment of a typical software-provisioning organisation. Demand of the software service provisioning business comprises of various kinds of software development projects such as, *low margin low risk* (LMLR), *medium margin low risk* (MMLR), *medium margin high risk* (MMHR) and *high margin high risk* (HMHR). The organisation bids for these projects and may have different win-to-bid ratios for different kinds of projects. A win-to-bid ratio signifies market perception of organisation's ability to deliver the given kind of project on time and with the desired quality. This is largely determined by track record of the

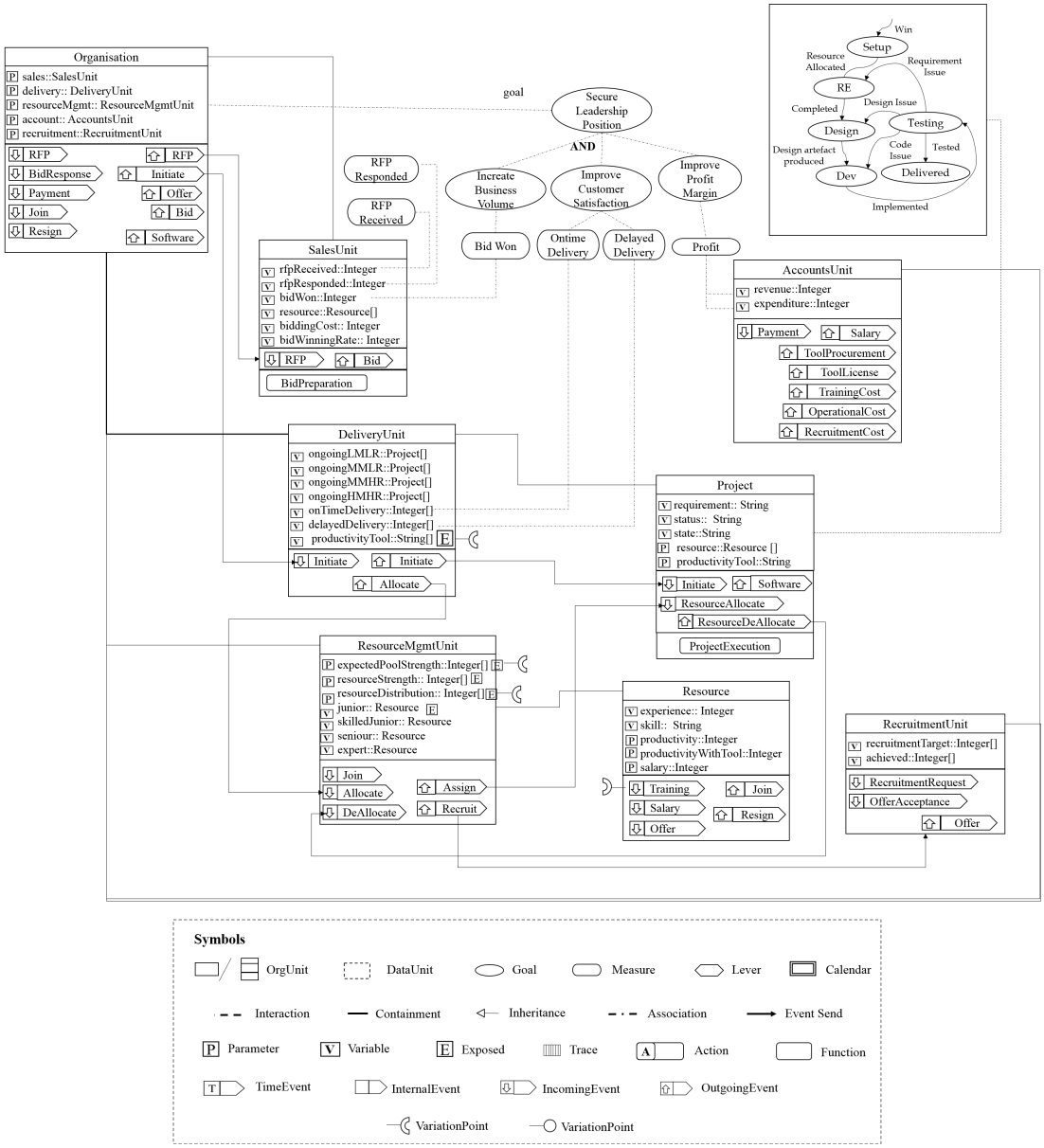


Figure 7.3 Internal structure of Software Service Provisioning Organisation

organisation. Supply is comprised of different kinds of workforce resources such as: *junior* (J), *skilled junior* (SJ), *senior* (S) and *expert* (E). Different kinds of projects may need different mix of resources. For instance, execution of HMHR project demands larger proportion of experts than, say, LMLR project.

7.1.2 OrgML model

The SSPO problem entity is modelled using three interacting `OrgUnits` namely: *Customer*, *Supply*, *Organisation*. The *Supply* `OrgUnit` is pools of junior, skilled junior, senior and expert

Resources which are also represented as `OrgUnits` as shown in Figure 7.2. The key interactions between *Customer* and *Organisation* `OrgUnits` are:

- *Customer* sends '*RFP*' `Events` for LMLR, MMLR, MMHR, HMHR kinds of software project to *Organisation*.
- *Organisation* `OrgUnit` prepares '*Bid*' and responds back to the sender *Customer*.
- *Customer* evaluates '*Bid*' (or a set of '*Bids*') and communicates '*BidResponse*' to the responded *Organisations*.
- *Organisation* executes a project and delivers '*Software*' to the *Customer* if a '*Bid*' is won by the *Organisation*.
- *Customer* pays '*Payment*' when '*Software*' is delivered and quality criteria is met.

In order to continue BAU operation, an *Organisation* recruits junior, skilled junior, senior and expert *Resources* from *Supply* pools through job '*Offer*'. *Resources* may '*Join*' or ignore an offer based on the joining rate of the *Supply*. The *Resources* can '*Resign*' from an organisation (conforming to the attrition rate of *Supply*).

The modelled SSPO *Organisation* is structured using five autonomous organisational units or `OrgUnits`: *SalesUnit*, *DeliveryUnit*, *ResourcesMgmtUnit*, *AccountsUnit* and *RecruitmentUnit* as shown in Figure 7.3. The *SalesUnit* is responsible for analysing customer '*RFP*', preparing '*Bid*' and sending '*Bid*' to the *Customers*. Internally, the *Organisation* `OrgUnit` delegates '*RFP*' to *SalesUnit* `OrgUnit` and *SalesUnit* `OrgUnit` sends '*Bid*' to the *Customer* `OrgUnit` on behalf of *Organisation* `OrgUnit`. Similarly, *Organisation* delegates '*BidResponse*' to *DeliveryUnit* `OrgUnit` to project initiation and execution when a project is won. *DeliveryUnit* initiates a *Project* by instantiating and parameterising *Project* `OrgUnit` (*i.e.*, LMLR, MMLR, MMHR, HMHR kinds of project) and allocating appropriate *Resources* from *ResourcesMgmtUnit* based on recommended resources distribution (*i.e.*, J:SJ:S:E distribution). The instance of a *Project* `OrgUnit` executes a project by following a standard software development process as depicted using a state-machine in Figure 7.3 and delivers *Software* to the customers. A project execution can be delayed if *Resources* '*Resign*' from the *Project* and/or *Resources* work less than their expected productivities. The *DeliveryUnit* may procure and use productivity tools, such as code generator, to increase the change of timely delivery.

ResourcesMgmtUnit `OrgUnit` manages the internal *Resource* pools (*i.e.*, J:SJ:S:E distribution). It allocates appropriate *Resources* to newly started *Projects*, tries to maintain a steady *Resource* pools for upcoming projects, and provides recruitment requirements to the *RecruitmentUnit* (using '*Recruit*' event). *RecruitmentUnit* consolidates recruitment requirements and makes offer to *Supply* resource pool by considering joining joining rate and attrition rate. The *AccountsUnit* keeps track of the financial aspects. The *Organisation OrgUnit* delegates '*Payment*' to *AccountsUnit* `OrgUnit`. The payment amount contributes to the '*revenue*' of the organisation. The expenditures come from the salary of the resources, tool procurement cost, tool licensing cost, resource training cost, recruitment cost and other operational cost (such as project setup cost) for which *AccountsUnit* maintains the record.

The goal, measures and levers (*i.e.* GM–L structure) of *Organisation* are also highlighted in Figure 7.3. As shown in the figure, the *Organisation* has a goal to '*Secure Leadership Position*' in software service provisioning space and it attempts to realise its goal using three sub-goals: '*Increase Business Volume*', '*Improve Customer Satisfaction*' and '*Improve Profit Margin*'. The business volume of the *Organisation* can be measured by the number of bids won (*i.e.* the value of '*bidWon*' `Variable` of *SalesUnit* `OrgUnit`), the customer satisfaction can be measured by two `Measures`: '*Ontime Delivery*' and '*Delayed Delivery*' where the *Measure* '*Ontime Delivery*' is associated with'*ontimeDelivery*' `Variable` of *DeliveryUnit* `OrgUnit` and *Measure* '*Delayed Delivery*' is associated with'*delayedDelivery*' `Variable` of *DeliveryUnit* `OrgUnit`. In this context, '*Ontime Delivery*' and '*DelayedDelivery*' respectively positively and negatively influence the customer satisfaction. The goal '*Improve Profit Margin*' can be measured using '*Profit*', *i.e.*, the difference between '*revenue*' and '*expenditure*' of *AccountsUnit* `OrgUnit`.

This OrgML model can be configured for a specific software service provisioning organisation by specifying parametric variables that include:

- Customer configuration:
  - Project arrival rate, *i.e.*, arrival rate of LMLR, MMLR, MMHR, HMHR kinds of project (number per month)
  - Bid winning rate for LMLR, MMLR, MMHR, HMHR (in percentage)
- Supply Configuration:
  - Joining probability for J, SJ, S, E (in percentage)
  - Attrition rate of J, SJ, S, E (in percentage)

Customer Configuration			
Project	Arrival Rate	Chargeable Rate (K USD/KLOC code)	Winning Rate (%)
LMLR	10/Month	100	40
MMLR	8/Month	200	30
MMHR	8/Month	250	30
HMHR	6/Month	400	30

Supply Configuration			
Resource	Recourse Availability	Joining Probability (%)	Attrition Rate (%)
Junior	80	70	10
Skilled Junior	80	70	10
Senior	70	60	10
Expert	60	60	10

Organisation Configuration						
Resource	Productivity (COCOMO standard)	Compensation (K USD/Month)	Training and Recruitment Cost (K USD)	Distribution (%)		**Other Parameters**
Junior	0.9	7	5	25	Resource Strength	1600
Skilled Junior	1	8	8	35	Expected Bench Strength	750
Senior	1	12	10	20	Productivity Tool Used	No
Expert	1.1	15	10	20	Tool Cost	0

Figure 7.4 Input parameters of Software Service Provisioning Organisation case study

- Organisation Configuration:
  - Productivity of J, SJ, S, E kinds of resources
  - Productivity of J, SJ, S, E when productivity tool is used
  - Operating cost of J, SJ, S, E (in USD/month)
  - Recruitment cost of J, SJ, S, E (in USD/person)
  - Tool procurement cost (in USD)
  - Tool license cost (in USD/year)
  - Bidding cost for LMLR, MMLR, MMHR, HMHR (in USD/bid)
  - Project setup cost for LMLR, MMLR, MMHR, HMHR (in USD/project)
  - Total number of current resources
  - J:SJ:S:E distribution of current resources
  - Desired bench strength (in percentage)

7.1.3 Instantiation, simulation and decision making

Figure 7.4 depicts the configuration parameters of a software service provisioning organisation and its *Customer* and *Supply*. For instance, LMLR projects are charged at the rate of 100K USD/KLOC (Kilo lines of code), 40% of LMLR project bids result in wins with 10 projects arriving every month – and similarly for MMLR, MMHR and HMHR projects. From the *Supply* side, 70% of the selected junior resources join and 10% of existing junior resources resign from

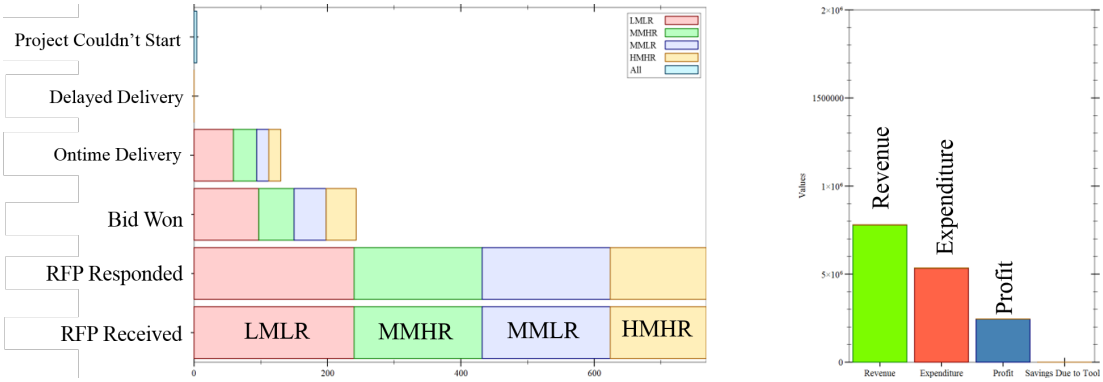


Figure 7.5 Simulation dashboard of Software Service Provisioning Organisation

organisation. Resignations do not necessarily happen in chunks, *i.e.* resources can resign any stage of a project execution but overall resignation count of a year is 10% of the total juniors. Initial resource strength is 1600. Within the organisation, 25% of available resource are juniors whereas the percentages for skilled junior, senior and expert resources respectively are 35, 20 and 20. Junior resources deliver a productivity of 0.9 (with respect to COCOMO standard [41]) and incur training and recruitment cost of 10K USD – and similarly for other kinds of resources.

Figure 7.5 shows the measures of the current-state. The horizontal histograms depict the number of RFP received, RFP responded, RFP won, projects completed on time, projects completed with delay and project pipeline (from to bottom to top), where four colors of each histogram (except project pipeline histogram) represent the metrics related to LMLR, MMLR, MMHR and HMHR respectively. The vertical histograms depict the revenue, expenditure, profit and saving due to productivity tool respectively (from left to right). The organisation is winning about 30% of the submitted bids and all of which are being executed within the expected time. Also, there is hardly any project that is not able to start due to non-availability of resources. Clearly, the organisation seems to be operating in a comfort zone.

Considering the organisational goal that aims to secure a leadership position, the management would like to explore possible options to improve the desired measures. The rest of this section explores a set of decision alternatives that include:

- **Exploration 1**: What is the best the organisation can achieve by removing existing slack?
- **Exploration 2**: What is the best the organisation can achieve with existing workforce distribution (J:SJ:S:ES)?

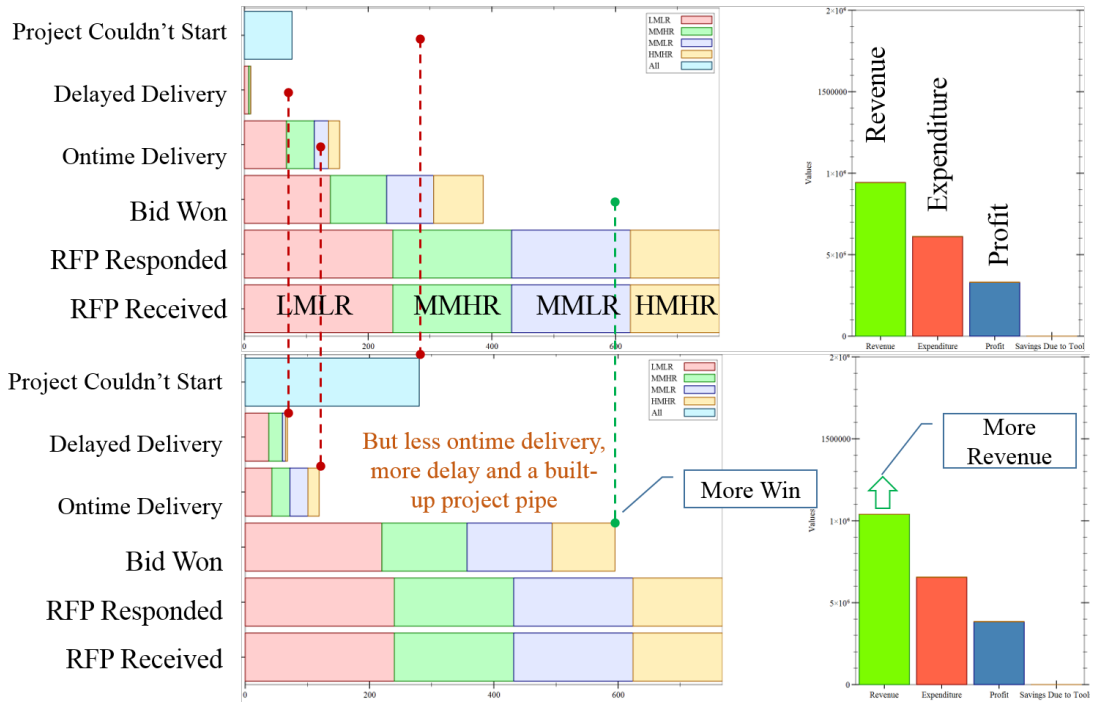


Figure 7.6 Effect of reducing price as well as delivery time

- **Exploration 3**: For these organisation settings, what is the best workforce distribution possible?

Exploration 1

The existing slack can be eliminated by winning more bids so that there are more projects to deliver. Delivery time and cost are the variables influencing bid winning percentage. Therefore, the chargeable rates for LMLR, MMLR, MMHR and HMHR projects are changed from 100, 200, 250, 400 (all in KUSD per KLOC) to 90, 180, 225, 350 thus improving their bid winning percentage from 40, 30, 30, 30 (all in percentage) to 60, 50, 50, 50. In addition, the promised delivery time for all the four kinds of projects are shorten by 1 normalised time unit thus further improving the bid winning percentage to 90, 70, 70, 70. Figure 7.6 shows the effect of this ajdustment on the organisation (*i.e.* applying a lever). The bid winning percentage improves to 65% from 30%. The number of projects completed on time remains more or less the same but there is a significant increase in the number of projects delivered late. Also, a significant number of projects witness delayed start due to non-availability of resources. Increase in bids won results in significantly high revenues even when chargeable price is reduced. With expenses

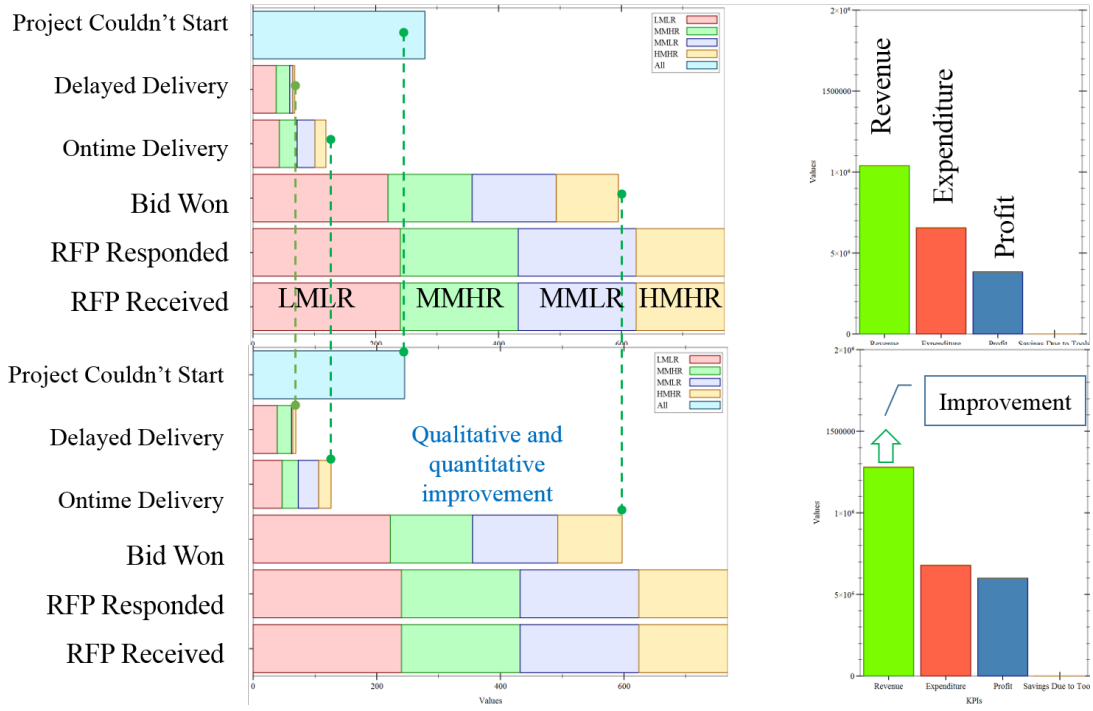


Figure 7.7 Effect of resource training

remaining more or less the same (linked largely to number of resources on board) profit is increased significantly.

Exploration 2

As seen from Figure 7.6, the delayed delivery and project kick-off queue build-up are critical concerns. How can these concerns be effectively addressed keeping the resource distribution unchanged *i.e.*, J:SJ:S:E::25:35:20:20? Clearly there is a need to increase workforce productivity. One can think of having a better-trained workforce or a better-tooled workforce or both. The productivity of junior, skilled junior, senior and expert are changed from 0.9, 1, 1, 1 (all as a factor of standard COCOMO productivity metric [41]) to 1, 1.1, 1.1, 1.1. This comes at increased training costs for the four kinds of workforce from 10, 10, 10, 15 to 20, 20, 20, 25. The improved measures are shown in Figure 7.7.

Productivity can be further increased by a factor of 1.25 by using productivity tools. This too comes at tool license and training costs. Figure 7.8 shows the effect of these changes on the measures. There is an increase in the number of projects delivered on time. More significantly, no HMHR project (shown in beige in Figure 7.8) is delivered late thus saving on delayed delivery penalties. Also, there is a significant increase in the proportion of HMHR projects delivered on

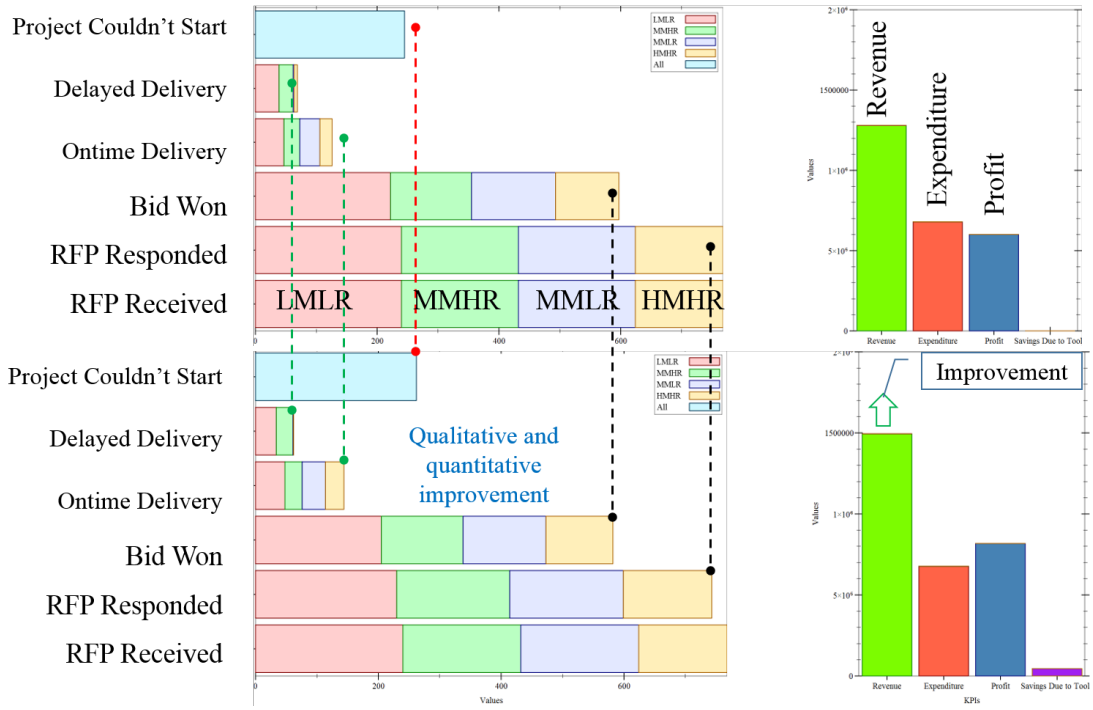


Figure 7.8 Effect of resource training as well as productivity tools

Figure 7.9 Allocation / deallocation trends of the four kinds of resources

time. The sum total of this is a significant increase in revenue as well as profits. However, the delayed start for projects remains a cause for concern.

Exploration 3

Figure 7.8 depicts the measures achievable with workforce distribution of J:SJ:S:E::25:35:20:20. But, is this the ideal workforce distribution for the prevailing demand and supply situation?

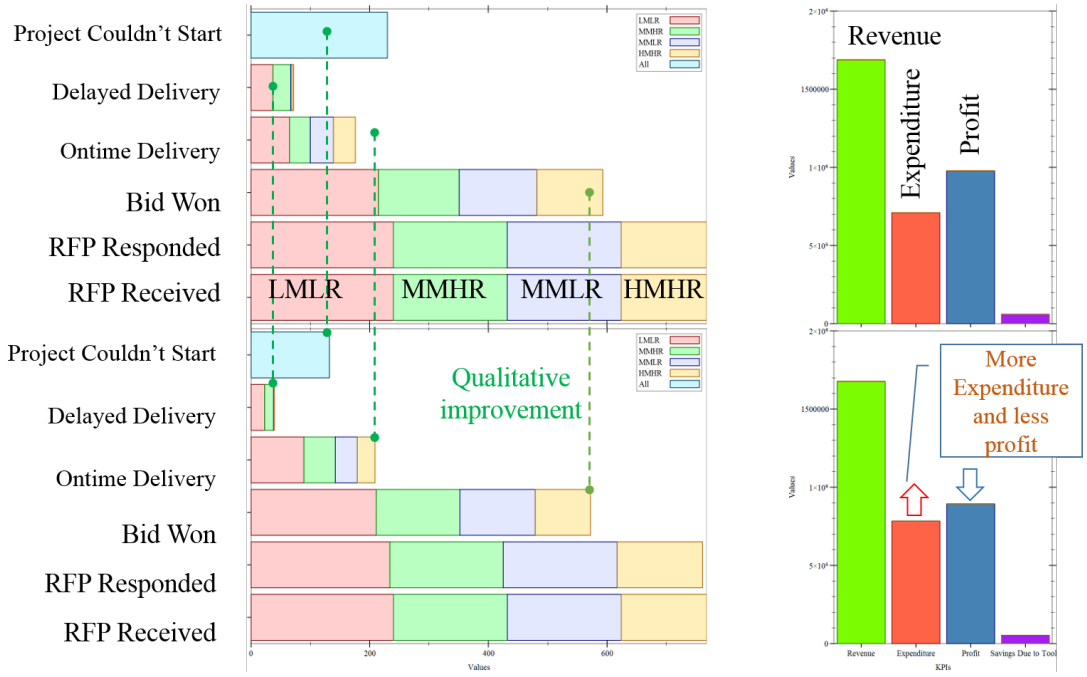


Figure 7.10 Effect of changed workforce distribution

Figure 7.9 shows allocation / deallocation behavior for the four kinds of resources over a given time period. Line graphs for junior, senior and expert are more or less similar with all ending above the desired bench strength level (at the termination of 24 Month simulation period). This can also be seen as an indication of their availability levels in the market. On the contrary, the line graph for skilled junior has a pronounced downward slope and ends up well below the desired bench strength level thus clearly vindicating the shortage of supply.

It is also clear from Figure 7.9 that a skilled junior is a critical resource for timely execution of projects and they are in short supply. Considering the resource availability, the best option is to experiment with proportion of seniors and experts in the workforce distribution. The workforce distribution of J:SJ:S:E::25:35:15:25 is considered. An increase in compensation of skilled juniors from 8 KUSD/Month to 10 KUSD/Month is considered, which is reflected in increase in joining probability (from 70% to 90%) and decrease in attrition rate (from 10% to 5%).

Figure 7.10 shows the effect of the changes. There is an increase in the number of projects delivered on time –especially HMHR projects – thus leading to enhanced revenue. However, the expenditure is increased due to the workforce distribution changed in favour of more expensive resources and also due to increase in pay-package for skilled junior. Still, increase in revenue is large enough to offset the increased expenditure thus resulting in increased profit.

7.1.4 Summary

This case study shows an ability to specify an organisation (*i.e.*, SSPO), organisational units (*e.g.*, sales unit, delivery units and accounts unit) that are hierarchically decomposed into finer units and sub units (using top-down decomposition), and dynamic structure, such as varying number of projects in a delivery unit, using `OrgUnits`. The case study also shows how the activity delegation of an organisation can be realised using `Events` delegation, the environments of an organisation (*i.e.* demand and supply) can be modelled using `OrgUnit` abstraction and possible `Levers` can be specified using a set of `Parameters`. It also shows a GM–L structure of SSPO, the `Measures` values, which are obtained from ESL simulation and visualised using OrgViz data visualiser, a set of what-if scenario playing leading the informed decision making.

7.2 Demonetisation

The cash in circulation in the Indian economy has steadily been increasing over the years. The total cash in circulation was 2.1 trillion rupees in 2001 and it reached 17.9 trillion rupees in early November 2016[2]. Uncontrolled cash flow in the system and a growing trend of cash-based transactions has led to a shadow economy. As a course correction, the Indian government demonetised the currency notes of 500 rupees and 1000 rupees[3]. Principally, 86% of the cash in circulation was pulled out from the system in a sudden announcement on November 8, 2016.

The initiative was implemented with several precautionary measures to avoid a financial crisis. For example, the ATM and Bank withdrawal limits were significantly reduced, and a limitation was imposed on the exchange of old notes wherein the citizens were allowed to exchange up to 4000 rupees with the remaining deposited to their bank account. In addition, the cash-less payment modes, such as mobile wallet and card payments, were incentivised. Despite all preventive measures, the demonetisation initiative resulted in prolonged cash shortages and several unforeseen situations. The government tried to ease the emerging situations through real-time monitoring and introduction of on-the-fly corrective measures. This reactive decision making approach led to the criticism that it had been poorly thought through and inadequately planned[4].

[2] https://data.gov.in/resources/statistics-notes-circulation-india-2001-2015/download
[3] https://en.wikipedia.org/wiki/2016_Indian_banknote_demonetisation
[4] https://hbr.org/2017/03/early-lessons-from-indias-demonetization-experiment

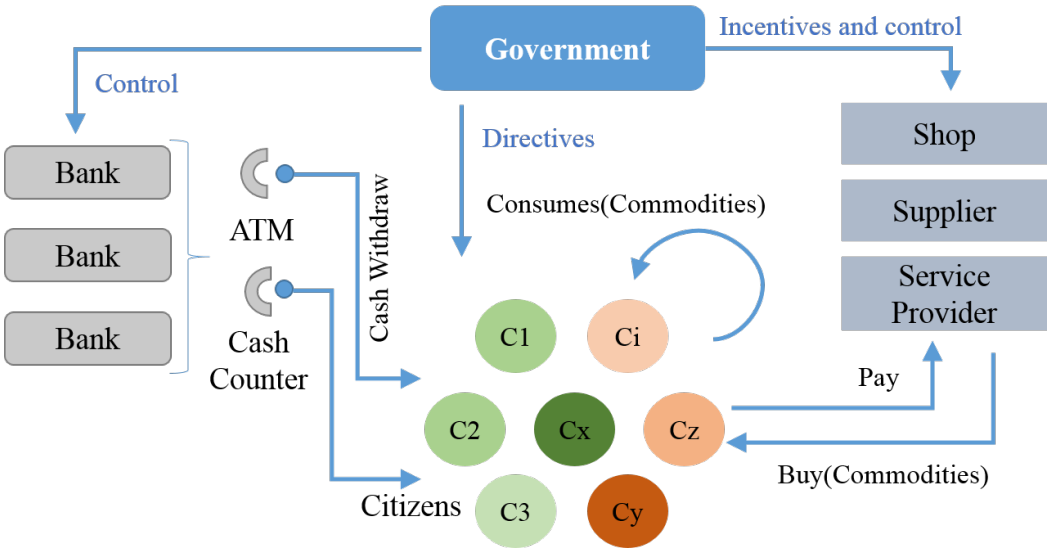


Figure 7.11 Pictorial representation of Indian Demonetisation scenario

7.2.1 Problem entity

The experiment considers a small but well-formed subset of demonetisation as a problem entity. The primary focus is limited to Indian citizens, who are largely confined to a bounded set of activities, as shown in Figure 7.11. Such citizens consume essential and/or luxury commodities (*e.g.*, food, medicines, cloths *etc.*), and use various services (*e.g.*, medical assistance, hospitality services, fitness related services *etc.*). A class of citizens may hold credit and/or debit cards – a citizen who holds card may choose to pay by cash or by card for a purchase, and may withdraw cash from ATM machine and/or bank counter. In contrast, a citizen without a card always pays by cash and withdraws cash from bank counters. This experiment assumes all citizens are able to satisfy their daily needs *i.e.*, poverty related societal conditions are excluded from this experiment.

The pre-demonetization stage is characterised by sufficient cash in ATMs and Banks to service their customers *(i.e.*, citizens), sufficient stock in shops, and no notable denial of service from banks and ATM machines (*i.e.*, citizens are able to withdraw cash when in need). This condition is considered a normal situation. The demonetisation event disrupts this by the abrupt elimination of 86% cash from the economy with a plan to slowly restore cash levels back to 70% of pre-demonetisation stage. Banks adapted several restrictions on cash withdrawals immediately after the demonetisation event to manage fair distribution of new currency notes being introduced at a fixed rate – a mint-centric constraint. Notable restrictions were: ATM withdrawal limit was reduced to rupees 2000 in a day for a citizen, bank withdrawal limit was

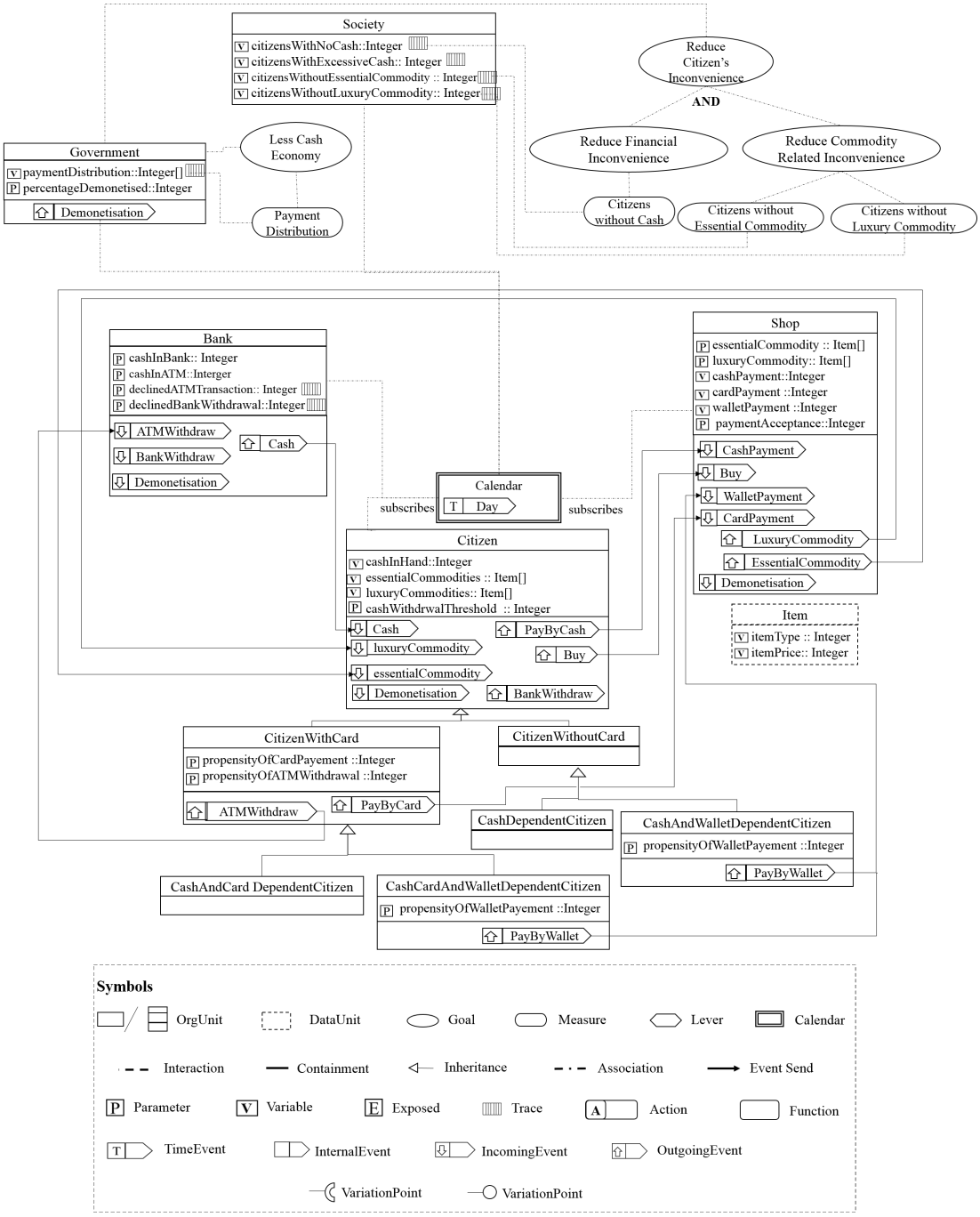


Figure 7.12 OrgML specification of Demonetisation case study

reduced to rupees 10,000 in a day for a citizen, and a weekly withdrawal limit was imposed to rupees 20000 per citizen. Shops adapted by accepting alternate payment options such as mobile wallet and card payment whenever they observed a drop in sales records. A citizen, as an individual, also adopted appropriate strategies to avoid undesired circumstances. The adaptation

strategies that were observed during post-demonetisation phase can be visualised along two dimensions:

- **Payment Pattern**: Citizens started using mobile wallet and/or card as a payment option to save the trouble of standing in long queues to withdraw cash. However, not everyone used alternate option, an individual's decision were based on several factors such as availability and familiarity with payment technology, and whether the citizen was an early or late adopter to the new technology.
- **Cash Withdrawal Pattern**: Some citizens resorted to temporary hoarding of cash *i.e.*, withdrawing cash in excess of their needs.

Given the above problem entity description, the experimental objectives are two-fold: (i) to understand if the normal condition is likely to be disturbed as a result of the disruptive change of demonetization, and to what extent, and (ii) to identify the courses of action to restore normal operation.

7.2.2 OrgML model

The problem entity is modelled using five autonomous and adaptive `OrgUnits` and an `Item DataUnit` where the `OrgUnits` are: *Bank*, *Shop*, *Government*, *Citizen* and *Society* as shown in Figure 7.12. The *Item* `DataUnit` is a representative entity for all kinds of essential and luxury commodities/merchandise/services; *Bank* `OrgUnit` represents a financial institution that stocks cash and allows citizens to withdraw cash through cash counters and ATM machines; *Shop* `OrgUnit` is an agent where *Items* can be purchased and services can be acquired; *Government* `OrgUnit` is an identity that observes situations and tries to control other identities; and *Citizen* `OrgUnit` represents common individual having a prototypical behaviour of which there could be many variants. A *Society* is visualises as a composite identity that comprises government, citizens, banks and shops. These `OrgUnits` are synchronised using a `TimeEvent` that represent day.

All primitive `OrgUnits`, such as *Bank*, *Shop*, *Government* and *Citizen*, react to the events of interest in a manner to help accomplish the goals as per a-priori known set of strategies as shown in Figure 7.12. The specification overview of the modelled `OrgUnits` are described below:

- **Citizen**: *Citizen* is modelled using a hierarchy of `OrgUnits`. Two kinds of *Citizens* are formed from the problem entity – a class of citizens hold card for financial transactions

(*i.e.*, *CitizenWithCard* `OrgUnit`), and other class of citizens are not having a card (*i.e.*, *CitizenWithoutCard* `OrgUnit`). Both kinds of citizens are further classified into two categories - (i) the citizens who can use wallet, and (ii) citizens who don't use wallet.

In general, the citizens store essential and luxury commodities for daily consumption, and they hold cash to purchase these commodities. The citizens typically have a cash threshold value that decides when they should approach bank/ATM to withdraw cash. These variables are modellled as '*essentialCommodities*', '*luxuryCommodities*', '*cashInHand*' and '*cashWithdrawalThreshold*'. A citizen can withdraw cash from bank, buy commodities from shops and pay for their purchases. These interactions are specified using '*BankWithdraw*', '*Buy*' and '*PayByCash*' `OutgoingEvents`. Similarly, the citizen receives cash when they withdraw cash from bank (and also from ATM) and receives commodities when they buy them from the shop. These interactions are modelled using `IncomingEvent`: '*Cash*', '*EssentialCommodity*' and '*LuxuryCommodity*'. The citizens who have a Debit/Credit card, which is represented as *CitizenWithCard*, may withdraw cash from ATM using '*ATMWithdraw*' and pay through card using '*PayByCard*'. Individuals have their preferences to use a card as opposed to cash as payment option, which is represented using '*propensityOfCardPayment*', and preference to use ATM as opposed to bank withdraw that is represented using '*propensityOfATMWithdraw*'. Similarly, the citizens who are willing to use wallet payment (*i.e.*, *CashCardAndWalletDependentCitizen* and *CashAndWalletDependentCitizen*) can pay by wallet (*i.e.*, '*PayByWallet*' event) and have their preference to pay by wallet as opposed to cash (and card), which is represented as *i.e.*, '*propensityOfWalletPayment*' variable.

The citizens consume items/commodities every day, buy them from the shops and pay to the shops, and withdraw cash from bank and ATM machines. The actions: item consumption, buying behaviours, cash withdrawal behaviour with and without cards are illustrated using the extended state machine notation in Figure 7.13 wherein the transitions with a firm line represent standard behaviour, firm line with single underlined label represent uncertain behaviour, firm line with double underlined label represent the impact of external uncertainty (*i.e.*, Demonetisation), dotted lines represent the behaviour adapted after demonetisation, and dotted line with single underlined label represent uncertain behaviour after adaptation respectively.

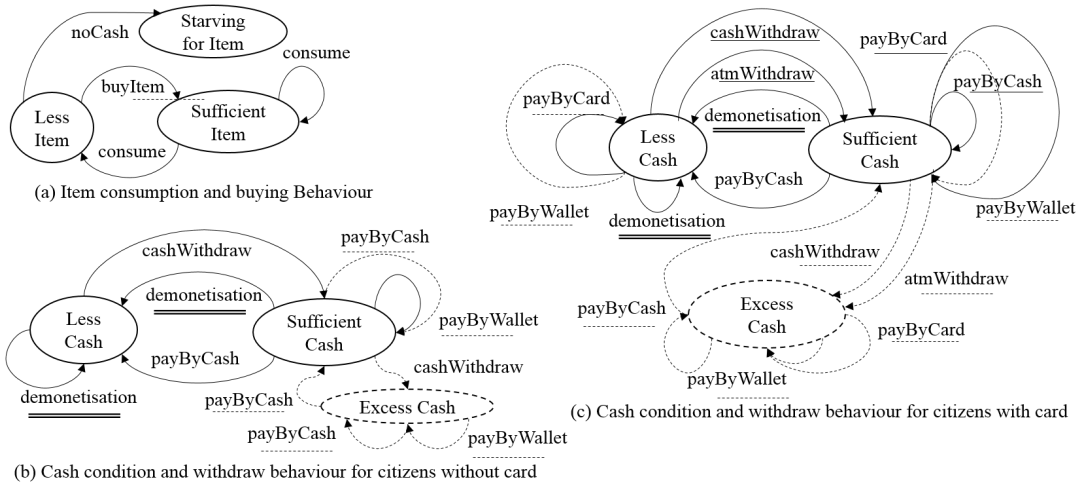


Figure 7.13 Behaviours of key OrgUnits

Figure 7.13 (a) describes item consumption and buying behaviour of an individual *Citizen*. A citizen can be in one of the three states: *Less Item* (item quantity dips below a threshold value), *Sufficient Item*, and *Starving for an Item* state. A *Citizen* consumes item to cater to daily needs; a consumption may change the state of a *Citizen*; *Citizen* attempts to buy *Item* when *Citizen* reaches to *Less Item* state; and a *Citizen* moves to *Starving for an Item* state from *Less Item* state if *Citizen* cannot buy an *Item* (due to *Less Cash* condition of other state machine such as Figure 7.13 (b) and 7.13 (c)). A *Citizen* can consume multiple *Items* as part of their daily life, thus a citizen may contain multiple state machines with varying states information for commodities being consumed.

Figure 7.13 (b) and (c) describe the cash condition and withdrawal behaviour of *Citizens* with and without cards. A *Citizen* with card may choose to pay by cash or by card for a purchase (a uncertain behaviour), and may withdraw cash from ATM machine or bank counter (another uncertain behaviour). In contrast, a *Citizen* without a card always pays by cash and withdraws cash from bank counters. The adaptation strategies of citizens are depicted using dotted lines in Figure 7.13 (b) and (c). A citizen, as an individual, may adopt an appropriate strategy (with multiple options selected based on personal intuition and experience – uncertain behaviours) to avoid entering an undesired state.

- **Bank**: *Bank* `OrgUnit` receives '*ATMWithdraw*' and '*BankWithdraw*' requests and dispense '*Cash*' (from bank counters or ATM machines) if cash is available *i.e.*,'*cashInBank*' or '*cashInATM*' holds a positive number. *Bank* maintains the trace of transaction declined statistics using '*declinedATMTransaction*' and '*declinedBankTransaction*' `Variables`.

Banks, typically, have three states *NoCash*, *LowCash*, and *WithCash*; they try to replenish cash when they are in *LowCash* state, and they refuse withdrawal requests when they are in *NoCash* state.

- **Shop**: *Shop* `OrgUnit` receives '*Buy*' request for essential and luxury commodities, delivers requested commodities using '*EssentialCommodity*' and '*LuxuryCommodity*' events, and receives payments through '*CashPayment*', '*CardPayment*' and '*WalletPayment*'. *Shops* maintain the records of cash, card and wallet payments using '*cashPayment*', '*cardPayment*' and '*walletPayment*' `Variables`.

- **Society**: *Society* `OrgUnit` contains all `OrgUnits` and monitors the financial and commodity related status of the citizens. In particular, it maintains a trace that shows how many citizens are without any cash in hand (*i.e.*, '*citizensWithNoCash*'), number of citizens who are hording excessive cash (*i.e.*, '*citizensWithExcessiveCash*'), number of citizens who have no essential commodities (*i.e.*, '*citizensWithoutEssentialCommodity*'), and number of citizens who have no luxury commodities (*i.e.*, '*citizensWithoutLuxuryCommodity*').

- **Government**: *Government* `OrgUnit` is a controller that monitors cash flow, initiates demonetisation, and observes payment distributions. The government has two goals - (i) reduce the cash flow from society, which can be measured using consolidated payment distribution patterns, and (ii) less inconvenience to the citizens. The citizen's inconvenience can be measured along two dimensions: financial inconvenience (*i.e.*, citizens without cash) and commodity related inconvenience (*i.e.*, citizens without essential commodities and/or luxury commodities) as shown in the OrgML model depicted in Figure 7.12.

In this setting, a society progresses with the primitive `TimeEvent` that represents a '*Day*'. Each day, citizen `OrgUnits` consume items, buy items from shops if any item is below a certain threshold, pay for the purchases, and make an attempt to withdraw cash if needed. Similarly, bank `OrgUnits` try to stock up cash to fulfill ATM and Bank withdrawal requests, and shop `OrgUnits` stock up the items for their customers (*i.e.*, citizens). The government `OrgUnits` triggers '*Demonetisation*' event at a specific day (an input parameter) of a simulation run. Overall, the citizens exhibit significant individualistic, uncertain, and adaptive behaviour with shops showing moderate dynamism whereas banks exhibiting largely deterministic behaviour.

Figure 7.14 Simulation dashboard of Demonetisation case study

7.2.3 Instantiation, simulation and decision making

In this experiment, a simulation run is separated into three phases: setup, pre-demonetisation, and post-demonetisation. Setup phase is an initial time span of a simulation run that derives values of the input parameters, which are dependent on the formation of a society. For example, the pre-demonetisation cash-flow rates and required cash stocks of the banks, required essential and luxury commodities at a specific shops are examples of such parameters. Pre-demonetisation phase is the time-span between setup phase and occurrence of demonetisation event. It is an observation phase that validates normal operation of a society that includes: (i) Banks have enough cash to service their customers; (ii) Shops have sufficient stock to cater to the needs of their customers; and (iii) Citizens face no problems in buying items as well as withdrawing cash. The post-demonetisation phase, in contrast, is an observation phase to understand the impacts of the demonetisation event.

The OrgViz Data visualiser is configured to show specified `Measures` and `Traces` as shown in Figure 7.14. Nine `Measures` are chosen to understand condition of a society at a specific time. The 'Citizen Type' table describes citizens and their card/wallet usage capabilities. 'Payment Distribution' pie chart shows distribution of Card (green), Wallet (blue) and Cash(red) payments. 'Payment Transaction Volume' chart describes the history of overall payment transactions where card transactions are displayed in green, wallet transactions in blue, and cash transactions in red. The 'Cash Availability in Bank and ATM' graph shows the history of cash availability at Banks

and ATMs using red and blue colours respectively. Similarly, the 'Transaction Declined Rate' graph describes the denial of service at Banks and ATMs using red and blue colours. In addition, 'Citizen with no Cash' and 'Citizen with excess Cash' charts describe the financial condition of the citizens: the former chart describes the number of citizens having considerably less cash, and the latter represents the number of citizens hoarding cash. The cash dependent citizens are displayed in red, cash and wallet dependent citizens in blue, cash and card dependent citizens in green, and citizen with all facilities in yellow. The 'Citizens without essential commodities' and 'Citizen facing inconvenience' charts represent the number of citizens starving for essential commodities and luxury commodities respectively.

For conducting experiments, a society with one government, one bank, 15 shops and 1710 citizen actors are simulated for 150 '*Days*', where the first 15 days are considered for setup phase, next 30 days are the pre-demonetisation phase, and 105 days are the phase to observe post-demonetisation effects. A snapshot of simulation dashboard at the day of 115 (*i.e.*, after 70 days of demonetisation) is depicted in Figure 7.14. As shown in the figure, the graphs are unstable for first 15 days of simulation run as the `OrgUnits` are trying to set the values based on the behaviours of other `OrgUnits` and their interactions. The simulation outcome for pre-demonetisation phase is stable and normal: no bank withdrawal request is denied, no citizen is facing any financial crisis, and citizens are not experiencing any deficiency for essential or luxury commodities. The '*Demonetisation*' event is triggered at day 45 causing a sudden reduction of 86% of all cash from the bank, shops and individual citizens. Subsequently, the withdrawals from bank and ATM decline whilst wallet payment and card payment increase significantly: the citizens have started facing a financial crisis and the citizens who are solely dependent on cash have started running short of essential and/or luxury commodities. The adverse effects continue for almost 50 days and then the situation returns to normal.

The graph with title 'Citizen with excess cash' in Figure 7.14 shows 115 citizens are hoarding cash when the situation is on the verge of returning back to normal. It is also observed that cash dependent citizens are more prone to cash hoarding behaviour. The 'Payment Transaction Volume' chart describing the history of overall payment transactions shows an interesting trend – the card (green) and wallet (blue) usage have increased in first 30–40 days of post-demonetisation phase, and then it slowly started reducing.

	Scenario	Cash Available in Bank and ATM	Transaction Distribution (<Cash, Wallet, Card> after 100 days)	No Denial of service at Bank and ATM (Days after Demonetisation)	Cash related Inconvenience		Commodity Related Inconvenience	
					Citizen with No Cash	Citizens with excessive Cash (After 105 days)	Citizens without essential item	Citizens without luxury items
1	Without Lever	Low	<50,42,8>	52 days	8.1% (45)	9.4%	7% (41)	26.3% (42)
2	Without Cash Hoarder	Low	<50,43,7>	40 days	7% (39)	0	6.1% (38)	25.7% (40)
3	With more Alternate Payments	Low	<35,57, 8>	45 days	7% (31)	9.4%	5.8% (34)	25.7% (35)
4	Reduced Bank and ATM withdrawal limits	Low	<50,45,5>	48 days	5.8% (46)	9.4%	4.7% (40)	23.4% (39)
5	Faster cash replenishment	Low	<70,15,15>	18 days	5.4% (17)	0	3.2% (16)	22% (15)
6	Combination of 2, 3, 4	Low	<20,45,35>	37 days	6.7% (35)	0	4.1% (26)	21% (29)

Figure 7.15 Simulation summary of Demonetisation case study

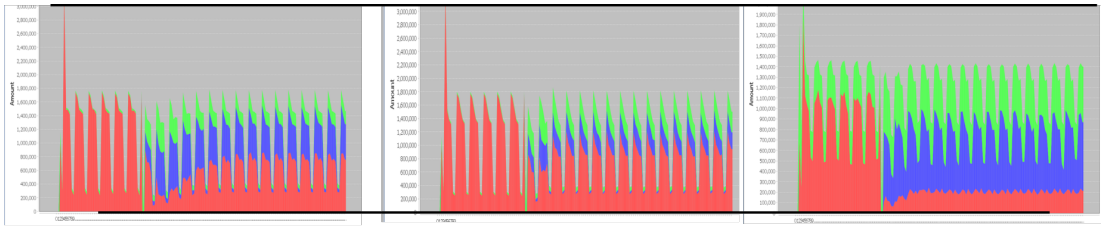
The `Measures` are correlated with the information found in authentic press-releases[5] and newspapers[6]. The trends on cash conditions of different citizens (shown in 'Citizen with no Cash' and 'Citizen with excess Cash' charts in Figure 7.14), the inconvenience due to deficiencies of essential commodities (shown in chart 'Citizens without essential commodities' in Figure 7.14) and luxury commodities (shown in chart 'Citizen facing inconvenience' in Figure 7.14) for cash dependent citizens, and service of denial at Bank and ATM withdrawal are in tune with the reality. In reality, the cash conditions in ATMs and Banks at the end of January 2017 (after 3 and half months of demonetisation) were just sufficient to serve their customers – this observation is consistent with the graph shown in 'Cash Availability in Bank and ATM' graph of Figure 7.14. Alternative payment volume trend 'Payment Transaction Volume' chart also matches with the Bloomberg report[7]. These observations and close correlations with reality ensure operation validity of the experimental model and simulation. After ensuring the operation validity, five what-if scenarios are developed either by modifying composition of society in terms of its constituent elements and/or modifying the characteristics of the constituent elements individually.

The scenarios and observed behaviours are summarised using a decision table in Figure 7.15. The row 1 is the standard configuration of a society, which is described above. Other five rows are the possible `Levers` that are explored as part of this experiment. The scenarios are: (i) a society without cash-hoarder citizen (row 2), (ii) a society with more e-wallet users – a case where citizens are convinced to use alternate payment options (row 3), (iii) reduced cash withdrawal limits where cash withdrawal limits from banks and ATMs were respectively reduced

[5]https://rbi.org.in/Scripts/BS_PressReleaseDisplay.aspx?prid=38520

[6]https://www.livemint.com/Industry/nhnU8KQPxP6y9FHrm8paUN/100-days-of-demonetisation-A-Mint-reading-list.html

[7]https://www.thequint.com/business/2017/02/17/demonetisation-100-days-indian-economy



(a): standard configuration (b): with faster cash replenishment (c): combined configuration (row 6)

Figure 7.16 Payment transaction volumes of Demonetisation case study

to rupees 1000 and rupees 5000 per day per citizen (row 4), (iv) faster cash replenishment where cash replenishment is 5 times faster than the standard configuration – a hypothetical case that is considered to know the situation if the government was well-equipped with newly minted cash (row 5), and (v) combination of the scenarios presented in rows 2, 3, and 4 of decision table shown in Figure 7.15.

Detailed simulation results[8] with operational graphics are not included in this section. Instead the results are summarised in Figure 7.15. The column 'No Denial of service at Bank and ATM' represents the day when denial of ATM and Bank withdrawal services are dipped below 5% in 'Transaction Declined Rate' graph (see Figure 7.14 as reference); column 'Citizen with No Cash' represents a tuple describing the peak value (*i.e.*, maximum number) of cashless citizens during post-demonetisation phase and time-span of 'Citizen with no Cash' graph (*see* Figure 7.14); and column 'Cash hoarder After 105 days' describes the number of citizens who are converted to cash hoarder at the end of simulation (captured from 'Citizen with excess Cash' graph). Similarly the columns 'Citizens without essential item' represents a tuple describing maximum number of citizens who were lack of essential items (from 'Citizens without essential commodities' chart) and time-span of such kind of inconvenience; and 'Citizens without luxury items' represents tuple describing maximum number of citizens who were lack of luxury items (from 'Citizen facing inconvenience') and time-span.

A comparative analysis of rows 1-4 of Figure 7.15 shows that the hoarding behaviour is one of the contributing factors for prolonged cash shortage – note row 2 is addressing the cash shortage issue better than other options. However, ATM and Bank withdrawal limits, as shown in row 4, are found as being critical to mitigate cash-less conditions and deficiencies of essential and luxury items – significant contributors to citizen inconvenience. This observation is in tune

[8] https://www.dropbox.com/s/q6xtz9el3sa6qzs/Demonetisation%20Experiment.pdf?dl=0

with the reality – the government had realized the importance of cash-limits after a week of demonetisation, and tried to arrive at optimum value through multiple alterations[9].

It was felt that faster introduction of new currency to banks and ATMs can lead to reduced inconvenience to the citizens. A simulation run with faster cash-replenishment (5 times more than standard configuration as shown in row 5) resulted in fewer cash shortages and less inconvenience to the citizens compared to other options. However, this option does not help to achieve less-cash society (as shown in column named as 'Transaction Distribution' of row 5). As cash was readily available in the desired quantity, citizens resorted to old habits *i.e.*, falling back on payments in cash at the exclusion of electronic payment options such as credit/debit cards and wallet payments as shown in the trace analysis depicted in Figure 7.16 (a) and (b) (where (a) is standard configuration and (b) is faster cash replenishment option respectively).

As part of exploring possible options that have the potential to reduce negative impacts of demonetisation while moving towards the less-cash society, an option that combines the options described in rows 2, 3 and 4 of Figure 7.15 is explored. The observed simulation results are recorded in row 6 of Figure 7.15 and Figure 7.16 (c). The result indicates significant improvement towards less-cash society as the alternate payment modes, *i.e.*, card and wallet transactions, in Figure 7.16 (c) are high as compare to Figure 7.16 (a) and 7.16 (b). The citizens without essential commodities and citizens without luxury commodities are also less as compare to the options depicted in rows 2, 3 and 4. Thus this experiment recommends a coordinated and judicious usage of multiple alternatives to achieve the specified goals.

7.2.4 Summary

This case study demonstrates four principal capabilities – (i) ability to form a system using a bottom-up approach. For instance, the society is formed by composing a set of citizens, banks and shops, (ii) emerging behaviour: the behaviour of the society is not specified but emerges from the micro-behaviours of the individual elements and their interactions, (iii) adaptability: citizens, shops and banks adapt to a new set of behaviours based on their conditions (*i.e.* expression of state variables) to represent post-demonetisation phase, and (iv) inheritance relationships to capture only cash dependent citizens, cash and wallet dependent citizens, cash and card dependent citizens and the citizens who use all options.

[9] https://en.wikipedia.org/wiki/2016_Indian_banknote_demonetisation

Table 7.2 Activities of Academics and Students

Key Element	Activities
Research Academic	Research, Paper Writing, Managerial Work, Unplanned Work (Query Resolution, Complain Resolution)
Teaching Academic	Prepare for Lecture, Deliver Lecture, Prepare for Student Assessment, Assess Student, Unplanned Work (Query Resolution, Complain Resolution)
Research and Teaching Academic	The combination of the activities of Research Academic and Teaching Academic
Student	Attend Lecture, Self Study, Appear for Assessment, Raise Query, Raise Complaint

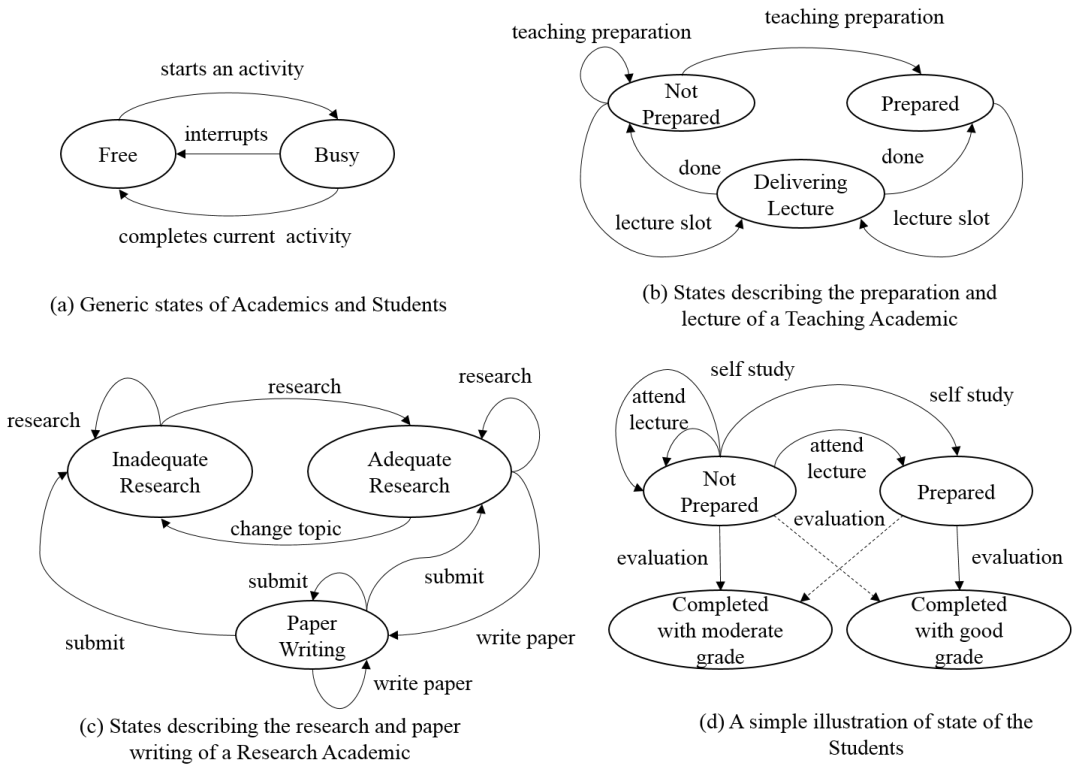
Consistent with the other case studies, this case study also demonstrates modularity, composability, reactiveness, autonomy, uncertainty and temporal behaviour. In addition, this case shows an illustration to understand the impact of a disruptive event on a complex socio-technical system using a synthetic environment. Precisely, a representative model is formed by mimicking well-defined micro-behaviors of the system elements, the fidelity of the model is established by correlating simulation results with real-life data collected from authentic sources, and the experimentation is conducted using actor-based bottom-up simulation approach to understand the impact of Indian demonetisation.

7.3 University case study

This case study considers a set of decision making scenarios of a department of ABC university used throughout the thesis as a running example. Consistent with the university goals, the department under consideration aims to improve its ranking by improving teaching quality and research outcomes. As discussed earlier that there are several courses of action or levers that can be explored to know which is the best suited for achieving departmental goals. This case study limits the explorations to (i) research and teaching academics distribution, (ii) define appropriate work preferences of the academics, and (iii) define suitable student/academic ratio.

7.3.1 Problem entity

Consider a department of ABC university that offers a set of courses to the undergraduate students and focuses on research activities. Every year, the students get enrolled on to the courses, teaching academics deliver a set of modules, and research academics make scholastic impacts through research. The teaching academics prepare and deliver lectures, prepare student assessments, evaluate students, and publish grades. The research academics undertake a range of



(a) Generic states of Academics and Students

(b) States describing the preparation and lecture of a Teaching Academic

(c) States describing the research and paper writing of a Research Academic

(d) A simple illustration of state of the Students

Figure 7.17 Behaviours of active elements of University

research activities that include conducting research work and publishing research papers. Some academics do both teaching and research. All such academics are responsible for clarifying student queries and resolving student complaints. These academics can work for a department on a part time or full time basis. Students attend lectures for their enrolled modules and appear for assessments to get grades. Students may raise queries in case of any doubt and they may raise complaints for longstanding unanswered concerns/queries. The typical activities of these individuals are highlighted in Table 7.2.

The generic behaviours of the academics and students are depicted using a simple state machine in Figure 7.17 (a). As shown in the state machine, an individual may start performing an activity from Table 7.2 for a specific time slot; from the `Free` state the individual moves to `Busy` state while performing the started activity and returns back to `Free` state at the end of it. In addition to this normal behaviour, an individual may suddenly interrupt an activity and returns back to `Free` state to initiate a high priority activity as shown in the state machine.

Precise micro-behaviours of a research academic, teaching academic and student are illustrated using non-deterministic state machines in Figure 7.17 (b), (c) and (d). As shown in Figure 7.17 (b), a teaching academic starts preparing for a lecture and eventually reaches

the Prepared state from Not Prepared state. Ideally, the teaching academics should deliver lectures when they are in Prepared state. However, they need to deliver lectures based on the department timetable (schedule), which is independent of the state of a teaching academic. Therefore a teaching academic may have to deliver a lecture either from Prepared state or from Not Prepared state. The teaching academic becomes busy and moves to Delivering Lecture state while delivering lecture, and they return to Prepared or Not Prepared state for the next lecture after delivering a lecture. The non-determinism and temporal uncertainty in the micro-behaviours presented in Figure 7.17 (b) are: (a) the transition from Not Prepared to Prepared state, and (b) the transition from Deliver Lecture state to Prepared or Not Prepared state.

The micro-behaviour presented in Figure 7.17 (c) describes the research and publication behaviour of a research academic. A research academic typically starts with Inadequate Research state when they are new to a research area. Eventually they move to Adequate Research state after researching on a topic for a specific period. From Adequate Research state, they may continue their research and stay at Adequate Research state, return back to Inadequate Research state by changing research topic or start writing papers by moving to Paper Writing state. From Paper Writing state, they have the following options - (i) stay at Paper Writing state and continue paper writing, (ii) return back to Adequate Research state and continue research work, or (iii) move to Inadequate Research when all research ideas are communicated and need further research for new publication.

The illustrative micro-behaviour of a student is presented Figure 7.17 (d). A student attends lectures and does self-study to prepare for modules. The evaluation of a module is conducted based on its schedule; therefore a student may have to appear for an evaluation from any of the two states: Not Prepared and Prepared. The propensity to reach Completed with good grade is high for a student if an evaluation is conducted on Prepared state, and Completed with moderate grade is high when the evaluation is conducted on Not Prepared state. However, there is a probability to reach any of the two states irrespective of their originating state as shown in the figure.

The overall behaviour of the department emerges from a set interacting micro-behaviours of the academics and students. Therefore, it is not possible to know the overall behaviour a-priori. The decision makers, such as a Dean and Head of Departments (HoD), judge the situations based on their experiences to explore decision alternatives such as: Is there any better distribution

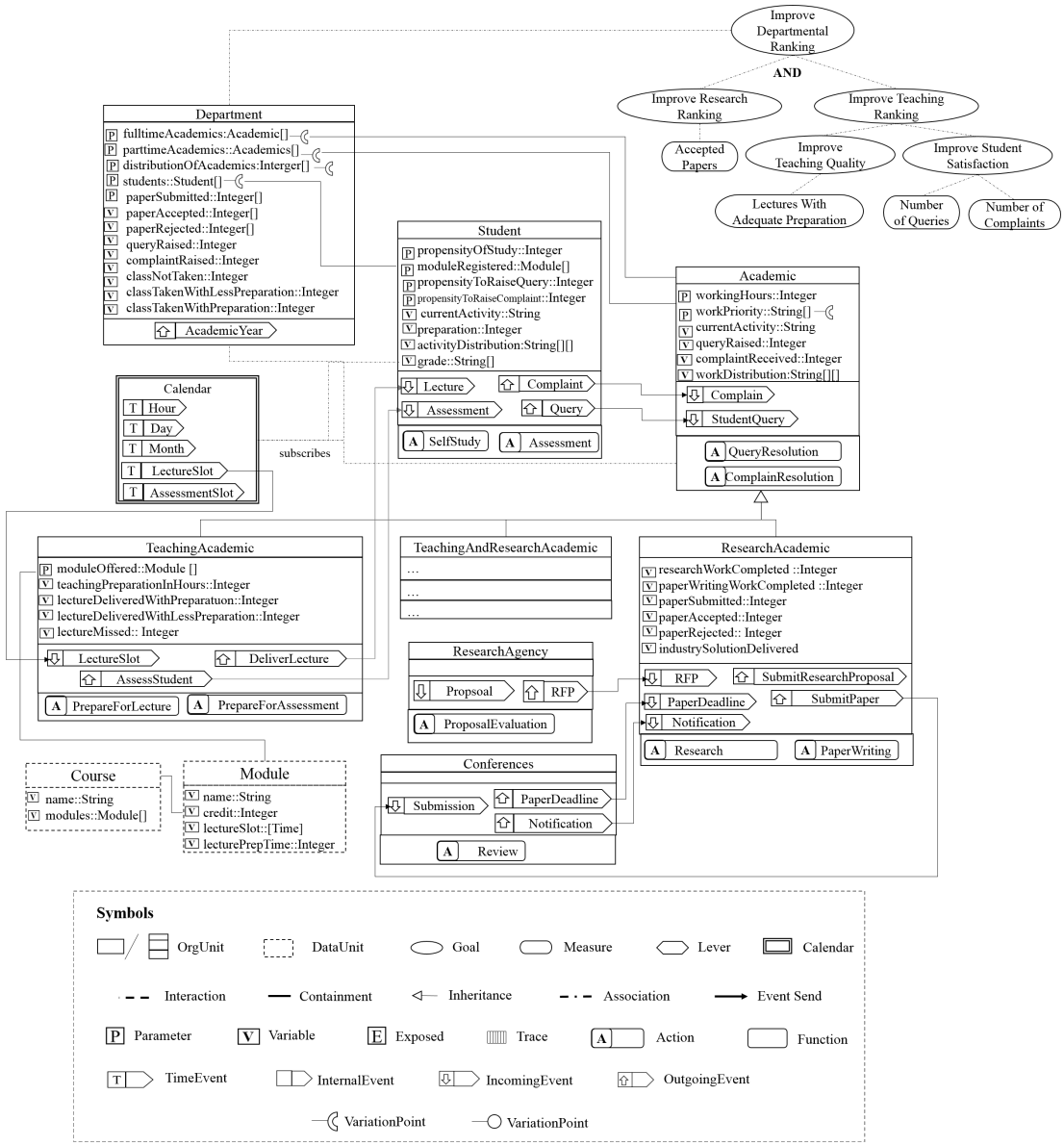


Figure 7.18 OrgML specification of University Department

of research and teaching academics that can improve the overall goal? Is the change in work priority of the academics useful to improve teaching and research qualities? Is a change in student/academic ratio useful for achieving goals? *etc*.

7.3.2 OrgML model

The above described problem entity is modelled using three primary `OrgUnits`: *Department*, *Academic* and *Student* to represent department, academics and students. The key elements of the OrgML model is depicted in Figure 7.18. The modelled `OrgUnits` are described below:

- **Academic**: Academic `OrgUnit` has a set of `Parameters` for characterisation and a set of `Variables` to capture its `State`. For examples, `Parameter` '*workingHours*' decides whether an academic is a full-time/part-time member of staff and '*workPriority*' describes the work priority of an academic (*i.e.*, preference to perform an activity from the list described in Table 7.2). The `Variable` '*currentActivity*' captures the state of an academic (*i.e.* 'Free' or 'Busy' with an activity), '*queryRaised*' captures the number of queries raised for the academic, '*complaintReceived*' describes the number of complaints received by the academic and '*workDistribution*' captures the activities performed by an academic (*i.e.*, hourly activities for working days/weeks/months). *Academic* `OrgUnit` receives '*Complaint*' and '*StudentQuery*' `IncomingEvents` and acts on them using '*QueryResolution*' and '*ComplaintResolution*' `Actions` respectively.

  The Academic `OrgUnit` is specialised into three sub-`OrgUnits` to represent teaching academics, research academics and the academics who focus both research and teaching. The TeachingAcademic `OrgUnit` captures offered modules using '*moduleOffered*' `Parameter` and keeps the records of teaching preparation hours, number of lectures delivered with adequate preparation, number of lectures delivered with less preparation and number of lectures missed using '*teachingPreparationInHours*', '*lectureDeliveredWithPreparation*', '*lectureDeliveredWithLessPreparation*' and '*lectureMissed*' `Variables`. From behavioural perspective, the *TeachingAcademic* subscribes `TimeEvents` specified in `Calendar`. It delivers lectures by raising '*DeliverLecture*' `OutgoingEvent` when they receive a '*LectureSlot*' and not busy with high priority activity. A *TeachingAcademic* misses a lecture if the academic is busy with high priority activity. In addition, the *TeachingAcademic* assesses students by raising the '*AssessStudent*' `OutgoingEvent`. Internally, *TeachingAcademic* prepares for lecture and student assessment using '*PrepareForLecture*', '*PrepareForAssessment*' `Actions`, which are triggered by internal events based on the '*workPriority*' and current state (*i.e.*, '*currentActivity*') of a *ResearchAcademic*.

  A *ResearchAcademic* keeps a record of research done (in hours), time spent on paper writing, number of paper submitted, number of paper accepted and number of paper rejected using '*researchWorkCompleted*', '*paperWritingWorkCompleted*', '*paperSubmitted*', '*paperAccepted*' and '*paperRejected*' `Variables`. It receives paper deadlines using '*PaperDeadline*' `IncomingEvent` and may submit papers using '*SubmitPaper*' `OutgoingEvent`. A *ResearchAcademic* researches on a specific topic using '*Research*'

`Action` and writes paper using '*WritePaper*' `Action`. They are triggered by the internal events based on the '*workPriority*' and '*currentActivity*' of a *ResearchAcademic*. The `OrgUnit` *TeachingAndResearchAcademic* is the composition of *ResearchAcademic* `OrgUnit` and *TeachingAcademic* `OrgUnit` definitions and their '*workPriority* can be defined based on the activities listed for *TeachingAcademic* and *ResearchAcademic* as shown in Table 7.2.

- **Conference**: The *Conference* `OrgUnit` is an aggregated and simplified model of conferences. It triggers '*PaperDeadline*' (along with call for paper) at a time interval, receives paper submissions (using '*Submission*' `IncomingEvent`, reviews them using '*Review* '`Action`' and sends '*Notification* '`OutgoingEvent`'.

- **ResearchAgency**: Similar to *Conference* `OrgUnit`, *ResearcgAgancy* `OrgUnit` is an aggregated and simplified model of the research funding agencies. It raises Request for Proposal (RFP), receives proposals, evaluates them and notifies evaluation outcome as shown in Figure 7.18.

- **Student**: *Student* `OrgUnit` is characterised by a set of `Parameters` that include '*propensityOfStudy*' (*i.e.*, probability of study when a student is in 'Free' state), '*moduleRegistered*', '*propensityToRaiseQuery*' (*i.e.*, probability of raising a query at a given time) and '*propensityToRaiseComplaint*' (*i.e.*, probability of raising complaint at given time). The state of a student is determined by a set of `Variables` such as '*currentActivity*' (*i.e.*, current activity of a student – it includes 'Free' state as well), '*preparation*' (*i.e.*, number of hours studied by a student), '*activityDistribution*' (*i.e.*, activities performed by a student) and the '*grades*' of a student.

  Behaviourally, a student may attend lectures when they receive a '*Lecture*' `IncomingEvent`. They appear for assessments by performing '*Assessment*' `Action` on an `IncomingEvent` named as '*Assessment*'. A student performs '*SelfStudy*' `Action` based on '*propensityOfStudy*'. Action '*SelfStudy*' updates '*preparation*' `Variable`. It may raise '*Query*' and '*Complaint*' `OutgoingEvents` based on '*propensityToRaiseQuery*' and '*propensityToRaiseComplaint*' `Parameters`.

- **Department**: *Department* is a composite `OrgUnit` that contains *Academics* and *Students* and consolidates individual `Measures` at the department level. As shown in Figure 7.18,

it contains a set of '*fulltimeAcademics*', '*parttimeAcademics*' and '*students*'. The distribution of *TeachingAcademic*, *ResearchAcademic* and *TeachingAndResearchAcademic* of the department is defined by '*distributionOfAcademics*' `Parameter`. Department specific consolidated values of the number of papers submitted, number of papers accepted, queries raised, complaints raised, classes/lectures not taken by the academics, number of lectures with less preparation and number of lectures with adequate preparations are respectively captured using '*paperAccepted*', '*paperRejected*', '*queryRaised*', '*complaintRaised*', '*classNotTaken*', '*classTakenWithLessPreparation*' and '*classTakenWithPreparation*' `Variables`.

The `Goals`, `Goal` decomposition structure and `Measures` of this case study are also shown in Figure 7.18. As depicted, the high-level goal of the modelled department is '*Improve Departmental Ranking*'. This top-level `Goal` is decomposed into two sub-goals: '*Improve Research Ranking*' and '*Improve Teaching Ranking*'. The goal '*Improve Research Ranking*' is a `LeafGoal` that maps to '*Accepted Papers* `Measure` wherein the `Measure` '*Accepted Papers* is linked to '*paperAccepted*' `Variable` of *Department* `OrgUnit`. On the other hand, `Goal` *Improve Teaching Ranking*' is decomposed into two `LeafGoals`: '*Improve Teaching Quality*' and '*Improve Student Satisfaction*'. The `LeafGoal` '*Improve Teaching Quality*' is mapped to `Measure` '*Lectures With Adequate Preparation*', which links to '*classTakenWithPreparation*' `Variable`. Similarly `LeafGoal` '*Improve Student Satisfaction*' is mapped to `Measure` '*Number of Complaints*' that links to '*complaintRaised*' `Variable` of *Department* `OrgUnit`.

Five `VariationPoints` are specified to describe `Levers`. The `VariationPoints` are:

- Number of full time academics (*i.e.*, '*fulltimeAcademics*' `Parameter` of *Department* `OrgUnit`)
- Number of part time academics (*i.e.*, '*parttimeAcademics*' `Parameter` of *Department* `OrgUnit`)
- Distribution of the academics (*i.e.*, '*distributionOfAcademic*' – ratio of *TeachingAcademics*, *ResearchAcademics* and *TeachingAndResearchAcademics* of the *Department* `OrgUnit`)
- Number of students (*i.e.*, '*students*' `Parameter` of *Department* `OrgUnit`)
- Work priority of the academics (*i.e.*, '*workPriority*' `Parameter` of *Academic* `OrgUnit`)

Figure 7.19 Input parameters of a Department

- **Course and Module**: *Course* and *Module* are modelled as `DataUnits`. Each *Course* is comprised of multiple *Modules* where a *Module* contains information about module credit, lecture slots, and typical preparation time for a well prepared lecture.

- **Calendar**: `Calendar` specifies five `TimeEvents` that represent: 'Hour' (a primitive time event), 'Day' (eight hours), 'Week' (five days), 'Month' (4 weeks), 'LectureSlot' and 'AssessmentSlot'. The 'LectureSlot' and 'AssessmentSlot' are a complex time expression as discussed in Figure 5.13 of Chapter 5.

7.3.3 Instantiation, simulation and decision making

For experimentation, the above OrgML model is instantiated for a *Department* named as '*CSE Department*' as shown in Figure 7.19. As depicted, the '*CSE Department*' is formed using 1200

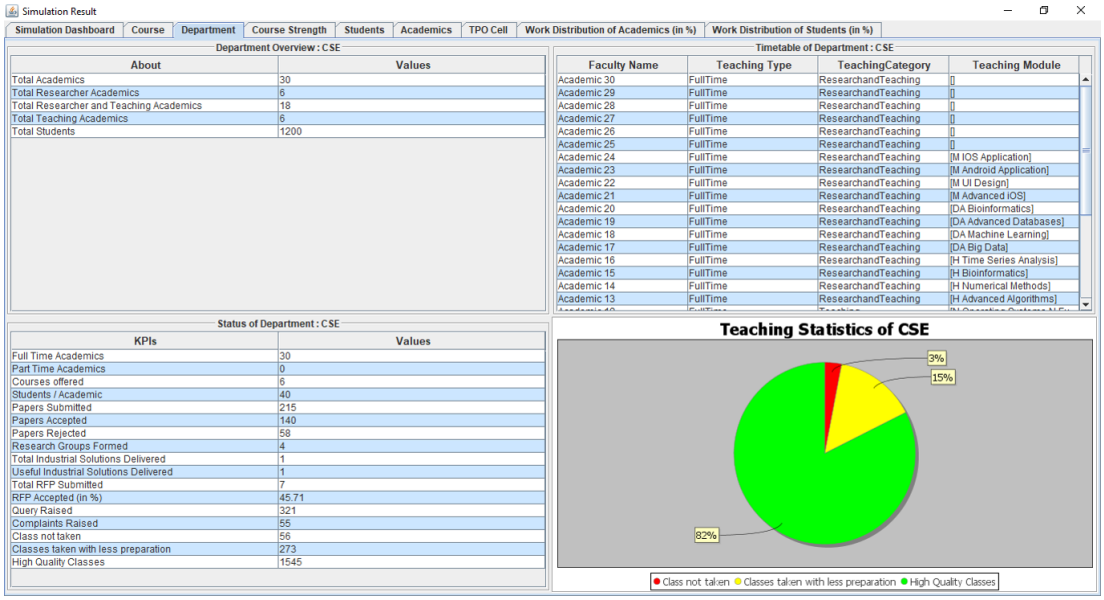


Figure 7.20 Simulation dashboard of University case study

Students and 30 full time *Academics* with 20:60:20 percent distribution of *TeachingAcademics*, *TeachingAndResearchAcademics* and *ResearchAcademics*. The work priorities of all three types of academics are shown in the figure. Further, the '*CSE Department*' is configured to offer six *Courses* where each *Course* is comprised of four *Modules*. The course names, their modules, and time slots of the modules are also shown using a set of tables in the figure. In this formulation, the instances of *TeachingAcademic* offer one or more *Modules* and they consider unplanned activities, such as addressing queries and complaint, as a high priority activity. The rest of the activities are prioritised as: teaching work, preparation for teaching and then assessment. On the other hand, the instances of *ResearchAcademic* prioritise their activities as: complaint resolution, research work, writing paper, other unplanned activities.

The configured and instantiated OrgML model is translated to ESL (using OrgML to ESL transformation rules), simulated for one '*Year*' using ESL engine and `Measures` are observed through OrgViz Data Visualiser. The simulation Dashboard showing an overview of the departmental `Measures` are shown in Figure 7.20. It shows the formulation of '*CSE Department*', offered *Modules* of the *TeachingAcademics* and the specified `Measures`. The Dashboard shows – 215 papers are submitted in a '*Year*', 140 papers got accepted, 58 paper got rejected, 321 queries are raised by the students, 55 complaints are raised by the students, a total of 55 lectures are missed by the academics, 273 lectures are delivered with less preparation

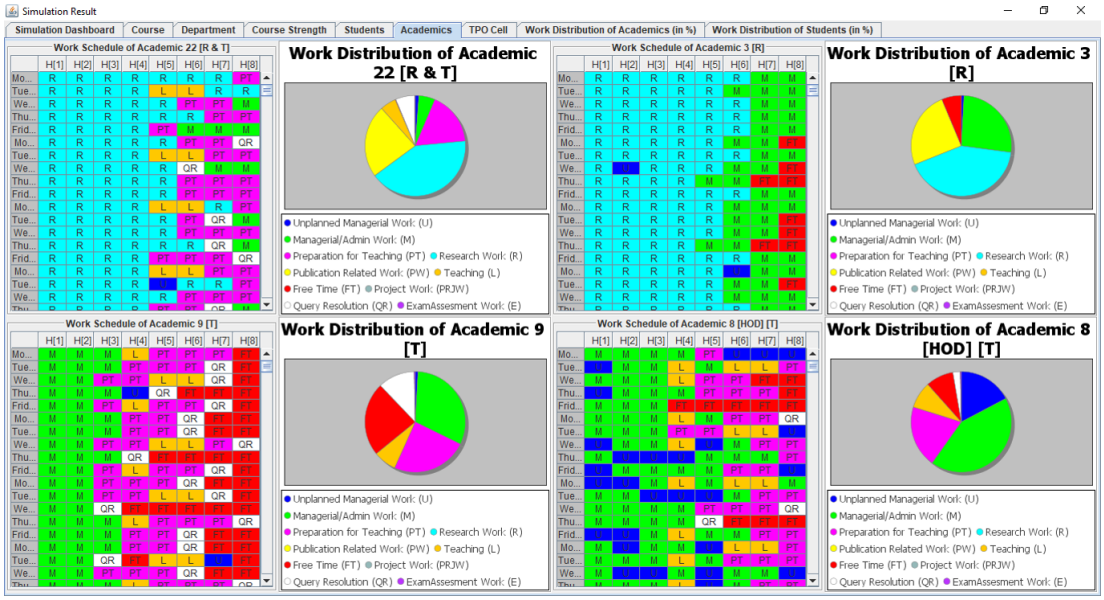


Figure 7.21 Status of Academics

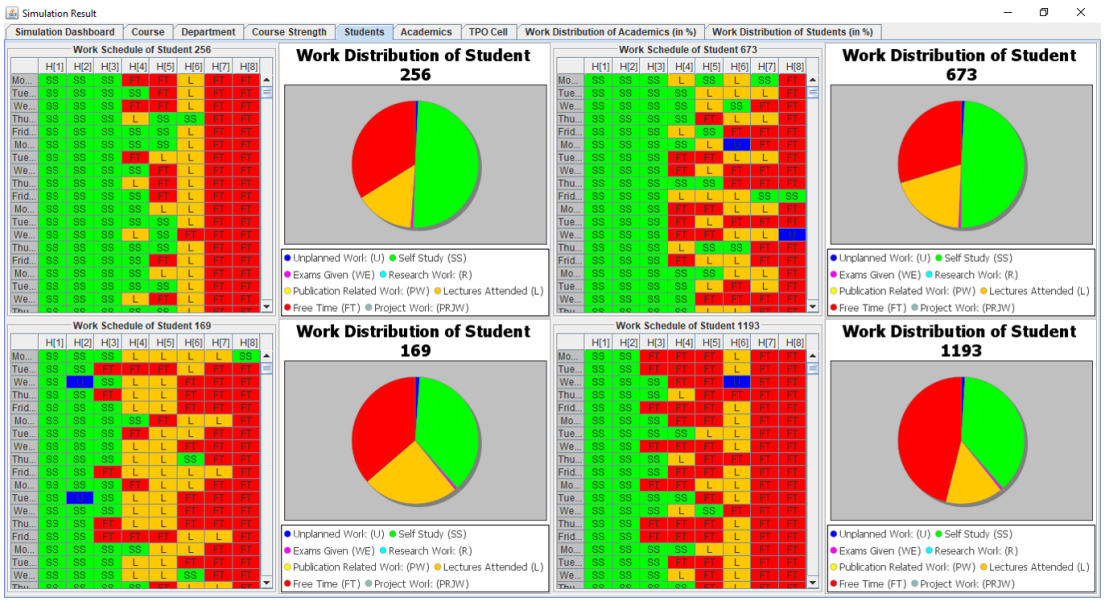


Figure 7.22 Status of Students

and 1545 lectures are delivered with adequate preparation. The overall teaching statistics of the department is also shown using a pie-chart in the figure.

The trace and latest snapshot of the work distribution of four (random) academics are shown using tables and pie-charts in Figure 7.21. Similarly, the trace and snapshot of activity distribution of four (random) students are shown in Figure 7.22. These graphs together show an indicative state of CSE department for the stated configuration.

	Levers	Full Time Academics	Students / Academic	Papers Submitted	Papers Accepted	Papers Rejected	Query Raised	Complains Raised	Class not taken	Classes taken with less preparation	Classes taken with adequate preparation
1	Initial Configuration<R = 20, R&T= 60, T = 20>	30	40	215	140	58	321	55	56	273	1545
2	Focus on Research <R = 40% , R&T=40%, T= 20%>	30	40	228	145	65	323	50	56	318	1527
3	Preference to Teaching Work for Teaching Academics.	30	40	224	114	90	212	62	64	153	1693
4	More Teaching Staff <R = 20% , R&T=50%, T= 30%>	30	40	185	124	43	181	45	49	178	1684
5	Improving Academics per Student Ratio	40	30	164	91	69	255	46	50	210	1632
6	Experiment with Ratio <R = 30% , R&T=35%, T= 35%>	30	40	246	156	62	157	51	58	107	1719

Figure 7.23 Consolidated simulation results

Now the decision questions are what will be state of CSE department (after one year) if it adopts different configurations, such as more research academics, more teaching academics, different distributions of research and teaching academics, different priorities of the academics or different students/academics ratio. These scenarios are explored by changing department formulation and observing simulation results. The observations are captured using a decision table as shown in Figure 7.23. The observations are: adding more research academics may not lead to better research outcomes (row 2). The change in work priority of the teaching academics (*i.e.* prioritising teaching activity as compared to assessment, query resolution and complain resolution) helps in improving teaching quality but negatively impacts the students satisfaction (row 3). Increase of teaching academics improves the student satisfaction but other goals are remain unaddressed (row 4). A better student/academic ratio (row 5) may not be an useful lever unless the additional academics are recruited appropriately as shown in Row 5. A distribution of teaching, research and teaching, research academics = <35, 35, 30>, as shown in row 6, produces most desirable outcome among the alternative experimented in this case study. This experiment provides an indication for further explorations. For instance, the decision table depicted in Figure 7.23 indicates a possibility to arrive at a better alternative by combining better distribution (as shown in row 6) and better work preference (as shown in row 3). Multiple such explorations help decision makers to arrive at a decision, which is backed by quantitative evidences. Moreover, these simulation explorations provide an indication of the positive and negative consequences of the alternatives before implementing them in reality.

7.3.4 Summary

This case study shows a scenario where a middle out modelling approach and combination of individualistic and aggregated behaviors are a pragmatic consideration for system specification and analysis. For example, the University can be decomposed into many departments and departments can further be segmented into research and teaching units by observing a University from a top-down perspective whereas the behaviour of a department or a research/teaching unit of the department can only be specified using behaviours of the academics and their interactions. Similarly, the individualistic behaviours of academics and students are necessary to describe and understand a department but the entities such as conferences, which are not the key elements for an analysis, can be aggregated for pragmatic realisation.

7.4 Evaluation

The previous three sections (and Appendix E) demonstrate the efficacy of the proposed OrgML based approach to simulate a range of what-if analyses and utility of OrgViz Data Visualiser to observe required measures for organisational decision making using four case studies with varying characteristics. For instance, the case study on SSPO illustrates how hierarchically decomposition structures (or a system of systems) of a fairly mechanistic organisation can be modelled and analysed using proposed approach. The case study on BPO organisation models an organisation as a monolithic *OrgUnit*. It demonstrates how an organisation and its competitors, which are competing with each other to achieve their goals in an uncertain environment, can be modelled and analysed. This case study shows a possible way to create an environment to analyse competition, collaboration and Nash-equilibrium point of a competitive business environment [18]. The case study on Demonetisation is a case where bottom-up modelling, probabilistic behaviours, adaptation, and emergent behaviour of a system are demonstrated. It also shows how a set of hypotheses can be evaluated using a simulation-based synthetic environment. Finally, the case study on University is formulated to demonstrate how the proposed approach supports autonomy and emergent behaviour, middle out modelling approach, and combination of individualistic and aggregated behaviours in a simulation setting.

The rest of this section evaluates the proposed research contributions along three dimensions – (i) the research artifacts are compared with respect to the state-of-the-practice and state-of-the-art technology aids to report improvements (along the requirements discussed in Table 3.3 of

Chapter 3), (ii) the applications of the proposed approach in different styles of organisational decision making (as highlighted in Chapter 3) are discussed, and (iii) the feedback received while research publications, tutorial presentation and industrial interactions are briefly summarised.

7.4.1 Comparison and improvements

For the kinds of decision-making problem illustrated in this chapter, industry practice mostly relies on spreadsheets, documents and diagrams. Such an approach captures the `Measures` and `Traces` of an organisation and specifies how they influence organisational `Goal` in terms of static algebraic equations. These equations are typically formulated based on the past observations and the experiences of the decision makers. The lack of support in expressing temporal behaviour, stochastic behaviours, adaptation and cyclic dependency over time limit the use of spreadsheets as data computation aid as opposed to a decision making tool. For example, the number of projects won in a month/quarter/year for given arrival rate of LMLR, MMLR, MMHR and HMHR of a Software Service Provisioning Organisation can be computed; the number of projects completed in a month/quarter/year can also be computed if number of resources are fixed; but predicting number of projects completed or whether a project is completed on time is not possible using spreadsheets when the number of resources changes in a project execution (due to attrition) or the productivity of individual resources differ over time (*e.g.*, more productive when they stay in project for longer period).

The inferential approach is often considered to solve organisational decision-making problems. It mines relevant historical data using data analytics and Artificial Intelligence (AI) techniques to identify the course of action that has resulted into best desirable outcome till date. The key concern of such approach is data adequacy and veracity. For instance, no historical data was available when Demonetisation initiative was implemented in India. Similarly, the data available in various Software Service Provisioning Organisations, Business Process Outsourcing orgaisations and Universities are often incomplete and fragmented. Moreover, the dynamic environment where they operate also changes over time, which makes the existing data less relevant for decision-making. For example, the bid evaluation strategies of Software Service Provisioning Organisation and Business Process Outsourcing organisation have changed significantly over time. They are much more competitive than the past. The cost arbitrage, quality of delivered software/service, the recognition from independent agencies are more important contemporary factors. Similarly the dynamics of the employee joining and leaving an organisation, their ability

to work as individual and with the productivity tools have also changed significantly over the years. Therefore, the strategy that worked in the past may not be an effective solution at present or in the future.

Another prominent approach is top-down modelling and simulation, such as Stock and Flow (SnF) [78]. An experimentation using multi-modelling and co-simulation that uses three prominent top-down EM techniques: i* [219], BPMN [209] and Stock and Flow (SnF) [78] is conducted on SSPO. The detailed experiment, models and simulation results are described in Appendix F. The goals of SSPO (as presented in Figure 7.3) is modelled using i* , the business process (*i.e.* the state-machine depicted in Figure 7.3) is specified using BPMN and overall dynamics of SSPO is specified using a SnF. In SnF model, the `Measures` are represented as *Stocks*, the factors that influence `Measures` are represented using *Flows*, the aggregated behaviour is encoded as equations over system variables that control the *Stocks* and *Flows*, and *Levers* are represented using auxiliary variables. The uncertainty and stochastic behaviour are specified using the probability distributions over stock variables and auxiliary variables of the SnF model. The temporal behaviour is captured by introducing appropriate time events and delays.

The experimentation is conducted using a tool-chain that includes OpenOME[10], Bizagi[11] and iThink[12]. As reported in [21], the what-if analyses exploring a set of `Levers` generate quantitative data to understand the long-term and short term implications of a set of `Levers` so that decision makers can take an informed decision. The issues which are observed from the experiments are:

- **Specifying Individualistic behaviour**: In an aggregated model, the individual characteristics get eliminated through averaging and the discrete events that may get triggered from multiple individual interactions are converted into probabilistic occurrences of events. Therefore, these models are somewhat removed from the reality and incapable of mimicking the emerging behaviour. For an instance, consider a junior resource of a software service organisation becomes a senior resource after a number of years. It is possible to capture this dynamism in a systems dynamic model but the impact of improved productivity and additional cost for this junior-to-senior transition in a specific project cannot be detected in a model that uses aggregation. Similarly, it is possible to determine the number

[10] www.cs.toronto.edu/km/openome
[11] https://www.bizagi.com
[12] https://www.iseesystems.com/store/products/ithink.aspx

Table 7.3 Technology advances

Requirements	EM Specs	Actor Lang.	ESL	OrgML	OrgML Concepts
Why	√	⊥	⊥	√	Goal
What	√	√	√	√	OrgUnit
How	√	√	√	√	Event and Behaviour
Who	√	⊥	⊥	√	OrgUnit
Where	√	⊥	⊥	⊥	OrgUnit
When	√	⊥	⊥	⊥	TimeEvent and Calendar
Modular	√	√	√	√	OrgUnit
Compositional	⊥	√	√	√	Composition relationship
Reactive	⊥	√	√	√	IncomingEvent, OutgoingEvent
Autonomous	×	√	√	√	InternalEvent and TimeEvent
Intentional	√	√	⊥	√	Goal
Adaptive	⊥	√	⊥	√	Adaptive Behaviour
Uncertainty	×	⊥	√	√	Stochastic Behaviour
Temporal	⊥	×	√	√	Temporal Behaviour
Measure Spec	×	×	×	√	Measure
Lever Spec	×	×	×	√	Lever
Top-down/ Bottom-up	Top-down	Bottom-up	Hybrid	Hybrid	Composition Relationship, Shared State Variable
√ : Supports adequately, ⊥: Can be specified with difficulties, ×: Not supported					

of projects completed over time given a certain joining and resigning characteristics, but determining the projects which are delayed due to attrition is not possible.

- **Specifying specialised behaviour**: Though a SnF model is not intended for specialised behaviour, it is possible to argue that it can be specialised for detailed analyses. But the effort required to specialise a SnF model at the level of types leads to model size explosion *e.g.*, specialisation of the notion of project into LMLR, MMLR, MMHR and HMHR projects and of the notion of resource into Junior, Skilled Junior, Senior and Expert resources in SSPO case study leads to 16 fold increase in model size. Similarly, introducing a new competitor with different parametric values also need additional SnF models.

- **Specifying localised behaviour**: SnF models offer poor support for modularity and change isolation. For example, incorporating a change in the bidding strategy of a specific kind of project (*e.g.*, the decision not to bid HMHR projects when significant project pipeline is built up) in SSPO case study may impact many flows and equations of SnF model. Similarly, the `Lever` definitions that involve localised structural and behavioural changes, such as different formulation of a *Department* or *Academics* with new set of

activities in University case study, require complete reformulation of the Stock-and-Flow model.

In contrast, the OrgML based approach enables modelling of a system of systems using a set of hierarchically composable autonomous `OrgUnits` each listening/responding/raising events of interest. Each individual system or `OrgUnit` encapsulates state (*i.e.*, a set of `State` variables), trace (*i.e.*, earlier states and events it has responded to and raised till now) and behaviour (*i.e.*, encoding of individual reactions). The behaviours and reactions are largely dependent on the state and trace of the `OrgUnit`. They interact with each other by sending messages resulting into emergent behaviour (*i.e.*, the behaviour of system of system emerges from interactions of `OrgUnits` or systems). Therefore, this thesis claims the proposed approach provides primitives for creating models that more closely mimic the reality.

The effectiveness of four approaches, *i.e.*, EM based approach (as discussed in [21, 120]), pure actor language based approach (such as [42]), ESL based approach (as presented in [121]), and proposed OrgML based approach are summarised in Table 7.3. As shown in the table, an EM based approach and an actor language based approach are complementary in nature. The former one supports aspect (*i.e.*, *why*, *what*, *how*, *etc.*) specification and a top-down simulation approach, whereas an actor language based approach is more effective for representing socio-technical characteristics and bottom-up simulation approach. But, it is not convenient for aspect specification. ESL is an improvement over actor languages as it supports uncertainty, temporal behaviour, and the bottom-up and top-down combination. However, ESL is a general purpose actor based simulation language and it is not convenient to specify *why*, *who* and *where* aspects and decision making constructs: `Goal`, `Measure` and `Lever`. Hence OrgML is further improvement as a specification language for organisational decision-making. It helps in expressing the most of the requirements in a convenient manner. Moreover, the proposed method helps to convert OrgML specification into a machine interpretable form and OrgViz Data Visualiser helps to visualise simulation data using user-specified visual forms for better sense making. Therefore, these capabilities collectively help decision makers to address complex dynamic organisational decision-making.

7.4.2 Applicability

The research artifacts produced in this thesis can be used as technology aids to address programmed decision-making, nonprogrammed decision-making and all four organisational

decision-making method templates or styles, which are discussed in Chapter 3. The programmed decision-making focuses on mechanistic organisations that are mostly governed by a fixed set of rules, which can be specified using *Action* specification proposed in this thesis. The nonprogrammed decision making considers system behaviours with significant uncertainty as discussed in all four case studies. Therefore, it is argued that both programmed and nonprogrammed decision-making can be addressed using proposed approach.

It can be argued that this approach can address all four organisational decision making styles – *Management Science* [8], *Carnegie model* [66], *Incremental Process model* [141] and *Garbage Can model* [60] (discussed in Chapter 3). The Management Science style considers precise `Goals` and explores finite and fixed set of `Levers` for mechanistic organisation. They are typically addressed using OR techniques. However, this approach can also be used for such problems as demonstrated in SSPO case study. In Carnegie model, the high level `Goals` are known and `Levers` are partially known as highlighted in BPO case study. For example, the primary goal of '*We*' organisation (described in BPO case study) is to secure a leadership position but expected quantitative values of the `Measures` to achieve its goals are not known as it is a competition environment. The BPO case study shows how this approach can be used for such kind of decision-making problems. Moreover, the Carnegie model recommends discussion, negotiation, coalition and bargaining to resolve conflicting beliefs about the possible consequences and arrive at a consensus decision [92]. It can be argued that such activities can be supported though specific *what-if* simulation and simulation results. In the Incremental process model, the decision-making starts with high-level *Goals* and fine-grained `Levers`. The effective `Levers` are typically identified/defined while experimenting with known set of fine-grained `Levers` in an incremental manner. In the Demonetisation case study, the less cash transactions is considered as one of the goal and the known set of `Levers` are to change in the micro behaviour of active elements, such as: citizen should not hoard cash and Bank should reduce cash withdrawal limit. The case study started with such micro-level changes and ended up with a `Lever` definition that combines multiple localised changes that need to be applied in a specific order (*e.g*, row 6 of Figure 7.15). The Garbage Can model or anarchy style that does not recommend a specific sequence, *i.e.*, `Goals` and `Levers` evolve and are evaluated simultaneously.

Table 7.4 Validation through Communications

Publications		
Research Activity	**Research Artifact Outcome**	**Conference &Journal**
Problem Identification	Research Problem and its relevant	Model Driven Engineering Languages and Systems (MoDELS) 2015, [RP12]
Literature Review	Systematic Literature Review of Enterprise Modelling (EM) techniques	Practice of Enterprise Modeling (PoEM 2016) [PR10]
Experimentation	Experimentation with existing modelling and multi-modelling techniques	Enterprise Modelling and Information Systems Architectures (EMISA) 2018, [RP2]
	Experimentation with various modelling and meta-modelling approaches	European Simulation and Modelling Conference (ESM) 2016, [PR9]
Conceptualisation	Hypotheses, conceptual model and requirements of organisational decision making	International Conference on Software Engineering and Applications (ICSOFT-EA) 2016, [RP11]
Research Contributions	OrgML meta model	European Simulation and Modelling Conference (ESM) 2017, [RP4]
	Model construction and decision making method	Practice of Enterprise Modeling (PoEM 2017) [PR5]
	OrgML based approach	MoDELSWARD 2017, [RP6],
	OrgML based approach and method	Model-Driven Engineering and Software Development – Spinger Book Chapter, [RP1]
Evaluation	Software Service Provisioning Organisation Case Study	European Simulation and Modelling Conference (ESM) 2017, [RP4]
	Business Process Outsourcing Case Study	European Modeling and Simulation Symposium (EMSS) 2017, [RP8]
	Demonetisation Case Study	Winter Simulation Conference (WSC) 2017, [RP3]
	University	Tutorials Practical Applications of Agents and Multi-Agent Systems (PAAMS) 2016 , Model Driven Engineering Languages and Systems (MoDELS) 2017

Doctoral Consortia

1. Practical Applications of Agents and Multi-Agent Systems (PAAMS) 2017, [DC1]
2. Practice of Enterprise Modeling (PoEM) 2016, [DC2]

Tutorials and tech briefing

1. Practical Applications of Agents and Multi-Agent Systems (PAAMS) 2016
2. Model Driven Engineering Languages and Systems (MoDELS) 2017
3. Innovations in Software Engineering Conference (ISEC) 2018
4. Model Driven Engineering Languages and Systems (MoDELS) 2018

7.4.3 Research artifact communications

Consistent with Design Science Research (DSR) guidelines (as highlighted in Table 2.2 of Chapter 2), the research artifacts and approach are reported in top-tier conferences, introduced as a new technological aid for organisational decision-making in multiple tutorial sessions in top conferences, presented in two doctoral consortia and discussed with several industrial practitioners. The research artifact validation through this list of publications, tutorials and doctoral consortia are summarised in Table 7.4.

7.4.4 Evaluation summary

This evaluation conforms to the research methodology described in Chapter 2. The *Artificial* and *Ex-Post* evaluations using four synthetic yet close to real life case studies and improvement analysis demonstrate the *efficacy* and *utility* of the research hypothesis and the proposed approach. Validations establish the usefulness of the research hypotheses that include the use of actor-based modelling abstraction and bottom-up simulation technique to capture complex organisation for decision-making. The case studies show the proposed concept model (presented in Chapter 3) and the requirements (described in Table 3.3 of Chapter 3) appear to be necessary and sufficient for organisational decision-making. OrgML meta-model (presented in Chapter 5) and OrgML specification language (presented in Chapter 6) are capable of representing the necessary information for organisational decision-making in an effective and integrated manner. Model transformation rules to translate OrgML into ESL (presented in Chapter 5) can produce specification for simulation-driven *what-if* analysis. Moreover, the proposed integrated and iterative method is useful for constructing organisational models and performing required *what-if* analyses in a systematic manner.

In addition to the *efficacy*, the proposed `OrgUnit` abstraction helps to improve the structural *clarity* [161] (*refer* Figure 2.3) of the organisational model and simulation-aided decision-making introduces an open-ended *learning capability* leading to organisational decision-making.

7.5 Limitations, threats and further improvements

While the proposed approach has been demonstrated to be an effective aid for a range of organisational decision-making problems, it does have limitations and it is vulnerable to certain threats. This section highlights the limitations of the proposed approach, discusses possible

threats to validity of the proposed approach and presents some improvement ideas as future work.

7.5.1 Limitations

The limitations of the proposed approach can be categorised into two broad classes – (i) specification and simulation related limitations and (ii) implementation related limitations.

From specification and simulation perspective, the proposed OrgML meta-model and specification language are not suitable for representing (and therefore analysing) geographical properties (*i.e.* spatial relationships). Secondly, the proposed approach is principally based on discrete-event simulation. Therefore, it is difficult to represent and analyse a system that relies on continuous-time. The other limitations, which can be addressed with less effort, are: (a) lack of advanced algebraic operators in OrgML specification to express complex equations, (b) inheritance and polymorphism in `DataUnit`, and and (c) support for *absolute* time.

As discussed in Chapter 6, the current proof-of-concept implementation realises a subset of the proposed approach. An industry scale implementation of the proposed approach and validation of the implementation using a real industrial scenario are considered as future work of this thesis. Industry scale implementation expects more advanced graphs and visualisation techniques in OrgViz Data Visualiser, the implementation of OrgML to ESL transformation engine needs to be evaluated using large examples, the implementation of *Goal Evaluator* and *Lever Recommender* modules are also critical to realise the full potential of the proposed approach.

7.5.2 Threats to validity

A threat to validity may arise when a new artifact is produced as a scientific contribution or new knowledge. This thesis proposes an approach to produce necessary quantitative information for informed organisational decision-making. Essentially, it produces new knowledge along two dimensions – (i) research contributions to address organisational decision-making (generic contributions for organisational decision-making problems) and (ii) quantitative information produced using the proposed approach to evaluate decision alternatives for a given decision problem (problem specific contribution).

The artifacts along both the dimensions are information system related artifacts. Therefore, they are vulnerable to four kinds of validities: *internal*, *construct*, *conclusion* and *external* as

Table 7.5 Approaches to address threats to validity of research contributions

Contribution	**DSR Artifact Category**	**Internal Validity**	**Construct Validity**	**Conclusion Validity**	**External Validity**
1. Conceptual model	Constructs	SMS along with DSR methodology	Broad SMS exploring a wide spectrum of decision styles, methods and techniques	Not relevant	Established generalisation from organisational theory to system theories
2. OrgML meta-model	Model	DSR methodology	Correlation with standard modelling, meta-modelling and conceptual modelling techniques	Not relevant	Not relevant
3. OrgML to ESL transformation	Method	Standard model to text (M2T) transformation technique	Transformation rules for all OrgML concepts	Not relevant	Not relevant
4. Overarching method	Method	DSR methodology	Established simulation method and state-of-the-practice decision-making methods	Not relevant	Not relevant

discussed by Gregor *et al.* [86] and Wohlin *et al.* [213]. The rest of this section discusses the threats to validity of two types of artifacts – (i) threats to validity of research contributions and (ii) threats to validity of the quantitative information produced using the proposed approach (and research contributions).

Threats to validity of research contributions

This research produces four primary contributions – Contribution 1: conceptual model to describe organisational decision making, Contribution 2: OrgML meta-model to capture relevant information for organisational decision-making, Contribution 3: a model to text transformation technique to transform OrgML model into simulatable ESL specification, and Contribution 4: an overarching method for model constructions, model validation and *what-if* analysis. The approaches adopted to address the threats to validity for the proposed contributions are summarised in Table 7.5.

The internal validity for *Constructs* artifact (*e.g.*, Contribution 1) is a concern when an inappropriate protocol is adopted for defining *Constructs* of a domain. This research uses a systematic mapping study (SMS) within an overarching and iterative DSR methodology for developing *Constructs* for organisational decision-making. The SMS helps to identify the

relevant concepts and their relationships. The overarching DSR methodology helps to validate identified concepts using multiple case studies . Moreover, it adopts a meta-modelling technique to represent identified constructs and their relationships in a structured form.

The threats due to construct validity may arise if *Constructs* artifact is produced based on inadequate information. This research explores a broad spectrum of management literature to define relevant concepts for organisational decision-making. The review includes – (i) all types of decision-making styles (*i.e.*, Management Science model, Carnegie Model, Incremental Process model and Garbage Can model), (ii) decision-making methods (such as the methods proposed by Richard Cyert [67], Herbert Simon [180, 182], and Richard Daft [69]), (iii) the key contributions in management literature from the year 1956 [67] to recent date [70], and (iv) industry reports such as McKinsey Quarterly [135] and Harvard Business Review [105] as discussed in Chapter 3. The conclusion validity arise due to incorrect interpretation of the available information, which is not relevant for this contribution as the management literature uses a descriptive approach to introduce and illustrate concepts. The external validity is relevant for Contribution 1 as the concept *organisation* is generalised to *systems*, *complex system* and *complex adaptive system* to understand characteristics of a complex organisation (as described in Section 3.1). However, the generalisations, which are considered in this thesis, are well established fact in management literature as discussed in [7, 10, 69, 183].

For OrgML meta-model (*i.e.*, Contribution 2), the internal validity is ascertained by adopting rigorous DSR methodology for conceptualisation, development and validation. The construct validity is ascertain by establishing correlation with conceptual modelling (as described in Section 5.2.1) and established modelling and meta-modelling techniques as described in Table 5.1. The conclusion validity is not relevant for OrgML meta-model as it is a novel contribution and the concepts of OrgML meta-model are not derived from any literature. Similarly, the external validity is not relevant for OrgML meta-model as it is a domain specific meta-model for organisational decision-making and the proposed meta-model is validated using multiple case studies from same domain, *i.e.*, organisational decision-making .

The internal validity of the proposed OrgML to ESL transformation (*i.e.*, Contribution 3) is ascertained by adopting standard model to text (M2T) transformation technique as described in Section 5.5. The construct validity is established by providing transformation rules for all concepts of OrgML meta-model (as shown in Table 5.2 and Appendix C). The conclusion validity is not relevant for this contribution as it is a novel contribution, which does not need

any interpretation of existing knowledge. Similarly, the external validity for OrgML to ESL transformation is not relevant as they are constructed for the proposed approach.

Finally, the internal validity of the proposed method (Contribution 4) is ascertained through DSR methodology. The construct validity is addressed by adopting well established simulation method proposed by Robert Sargent in [174] and the management viewpoint proposed by Richard Daft in [70]. The conclusion validity is not relevant as adopted methods are well documented and established methods in respective fields. Similarly the external validity is also not relevant as the adopted method are not generalised rather they perform the same set of tasks as recommended in respective methods.

Threats to validity of quantitative information produced using proposed approach

The validity analysis and threats to validity for quantitative artifacts in software engineering is proposed by Wohlin *et al.* [213] and Cook *et al.* [64]. In quantitative research, the conclusion validity aims to ensure the treatment or interpretation of an experiment is same as the actual outcome of the experiment. For the proposed approach, the conclusion validity is to ensure the simulation results are interpreted correctly by the decision makers. The OrgViz Data Visualiser provides a visual aid to the decision makers such that they can rely on simple descriptive statistics to interpret simulation results.

The internal validity is related to how well the experiment is done and the chance for confounding in an experiment. The adoption of an established process of simulation research suggested by Robert Sergent in [174] and conformance with the management guidelines discussed in Management literature, such as decision processes defined by Richard Daft [68] and Langley *et al.* in [123], establish the internal validity of an iterative *what-if* scenario playing.

Construct validity defines how well an experiment measures up to its claims. In the proposed approach, the construct validity is the closeness of the constructed model with the problem entity or real organisation. The construct validity is ascertained through methodological support to perform state-of-the-art simulation model validation techniques proposed in [174]. Principally, the operational validity is adopted where the constructed model is simulated for known scenarios and the obtained simulation results are manually compared against real data to ascertain the model validity.

The external validity is necessary when a generalised inference from outside the scope of an experiment is considered in the experiment. But this research is not considering such external

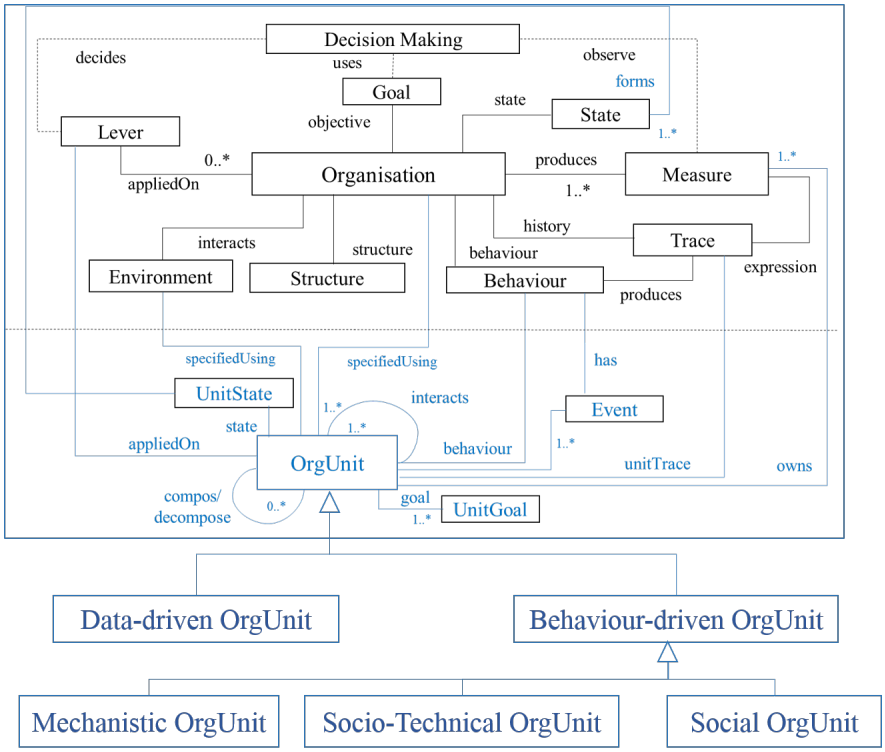


Figure 7.24 Extended conceptual model for organisational decision making

inference in a decision making. It is expected that a decision making problem should start with a problem entity. The construction of a purposive conceptual model, construction of simulation model and required *what-if* analyses should be performed for each decision making problem. Therefore, external validity is not relevant for quantitative information.

7.5.3 Further improvements and future work

Improvement opportunities are identified while evaluating research artifacts. The proposed approach considers OrgML as the conceptual specification and ESL *actor* as the simulation specification for `OrgUnit`. However, there can be some OrgUnits in an organisation, which can be conveniently specified using algebraic equations, Stock-and-Flow model, linear programming or Bayesian networks instead of the current form of OrgML specification. Moreover, they can be analysed using suitable computing aids rather than ESL engine. Therefore, an OrgUnit centric modular specification and an extension to OrgUnit specification to support multi-modelling capability and enable a co-simulation environment to simulate them in a coordinated manner (using event as a means for coordination) can be considered. For an instance, the conceptual model of an organisation can extended by specialising the OrgUnit into a hierarchy of OrgUnit

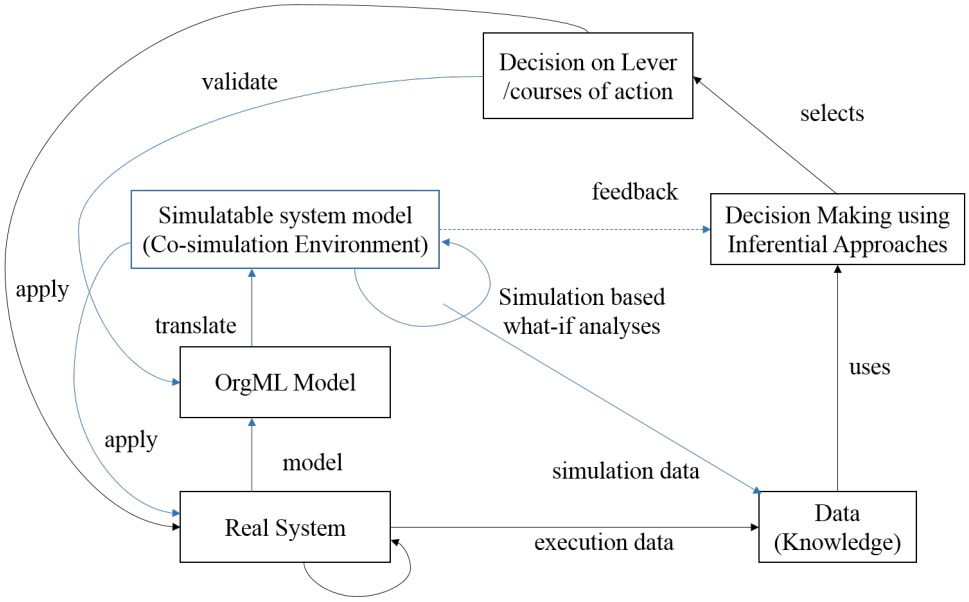


Figure 7.25 Integrated approach for organisational decision making

types as shown in Figure 7.24. Principally, the OrgUnit can be specialised into two kinds: *data-driven OrgUnit* and *behaviour-driven OrgUnit*. The organisational units whose behaviour is not precisely known but they behave/react/response based on their historical occurrences can be represented as data-driven OrgUnit. AI techniques are useful formalism to specify such OrgUnits. On other hand, the behaviour-driven OrgUnit can capture organisational units whose behaviours are known. They can be further classified into three classes: *mechanistic OrgUnit*, *socio-technical OrgUnit* and *social OrgUnit*. The mechanistic OrgUnit can be represented using OrgML. The linear programming [48] and Stock-and-Flow model [134] can also be considered to specify such OrgUnits if the aggregated analysis serve the purpose. The socio-technical OrgUnits can be best specified using OrgML specification. The social OrgUnit, in contrast, can be represented using specifications which are capable of representing *spatial* relations such as cellular automata [214]. The extension of OrgML specification to support *spatial* relations is another alternative to support all types of behaviour-driven OrgUnits.

Another way to improve the utility of the proposed approach is to use is in conjunction with the inferencing technique as presented in Figure 7.25. As shown in the figure, the inferential techniques use execution data to predict/select the best course of action or lever of an organisation. The key issue of such approaches is insufficient data and lack of fidelity of the existing data. The proposed approach can be used to produce missing data to make an inferential technique effective. In addition, an identified lever using an inferential technique can be verified (*i.e.* the

consequences can analysed and risk can be estimated) prior to its implementation using an OrgML based synthetic environment.

Several implementation specific improvements are identified in the research validation. For example, a graphical representation as opposed to current text-based specification for OrgML specification help to improve the *usability*[13] [161] of the proposed approach. OrgML notations (listed in Appendix B), which are used to represent four case studies, can be considered as an initial step to realise the graphical representation for OrgML specification.

Other improvements are the implementation Goal Evaluator and Lever Recommender. The statistical techniques for similarity analysis and distance measurements, such as multivariate distance matrix regression analysis [9], is a prospective exploration direction to realise an effective goal evaluator. Towards the Lever Recommender, the sensitivity analysis [171] of parametric values of a lever measures could be an effective method to identify sensitive lever. The use of machine learning techniques to extract meaningful patterns from simulation runs, *mutation* and *crossover* techniques of genetic algorithms [71] might be useful to arrive at a meaningful variant of an existing levers. The use of randomization techniques (*e.g.*, Monte Carlo simulation [144]) to derive manageable and meaningful set of random levers might be useful for decision space exploration. These improvements further reduce the burden of the decision makers. These opportunities along with the limitations associated with the proposed approach are considered as the future work to this thesis.

7.6 Summary

In this chapter, the proposed approach and research artifacts are evaluated using an *Artificial* and *Ex-Post* strategy. The evaluation demonstrates the efficacy of OrgML specification to capture a range of complex organisations. The case studies show how OrgML based approach helps decision makers to explore the decision space using quantitative evidences as opposed to intuitions. The improvements and applicability of the proposed approach are justified. The communications of the research artifacts are highlighted. The limitations of the proposed approach and research artifacts are discussed. The scope for improvements are discussed to widen the application scope of the proposed approach.

[13](DSR Classification :: Environment):: Consistency with people:: Ease of use :: usability [161]

Chapter 8

Conclusion

This thesis has set out the key challenges facing organisational decision makers. It reported on the issues of making an ineffective decision and its impact on opportunities in the future. The practice of organisational decision making today, centres around intuitions and past experiences. At best, the decision makers make use of computational tools, such as spreadsheets, to analyse historical data for future prediction. This research has argued that an intuition-driven approach is ineffective for modern organisations. Instead, organisational decision makers need advanced technology support to understand inherent dynamism, uncertainty, nonlinearity and emergent behaviour of the problem space.

The research conducted as part of this thesis, produced a series of rigourous literature reviews that addressed organisational decision-making, state-of-the-art enterprise modelling and analysis techniques, existing multi-modelling and co-simulation approaches, and actor/agent technologies. Organisational decision making was reviewed to help formulate the core problem statement and was done purely from the management and IS perspectives. Political, ethical, psychological, and social perspectives were considered out of scope for this thesis.

The systematic literature reviews identified several shortcomings: (i) an inability to capture necessary aspects of organisation and their inherent characteristics in an intuitive and domain specific manner (considering organisational decision-making as a domain of interest), (ii) inadequate analysis capabilities to precisely understand the socio-technical characteristics and emergent behaviour of organisation, and (iii) lack of a method to effectively use technology aids (as management view of organisation decision-making is ignorant of effective utilisation of technology aids).

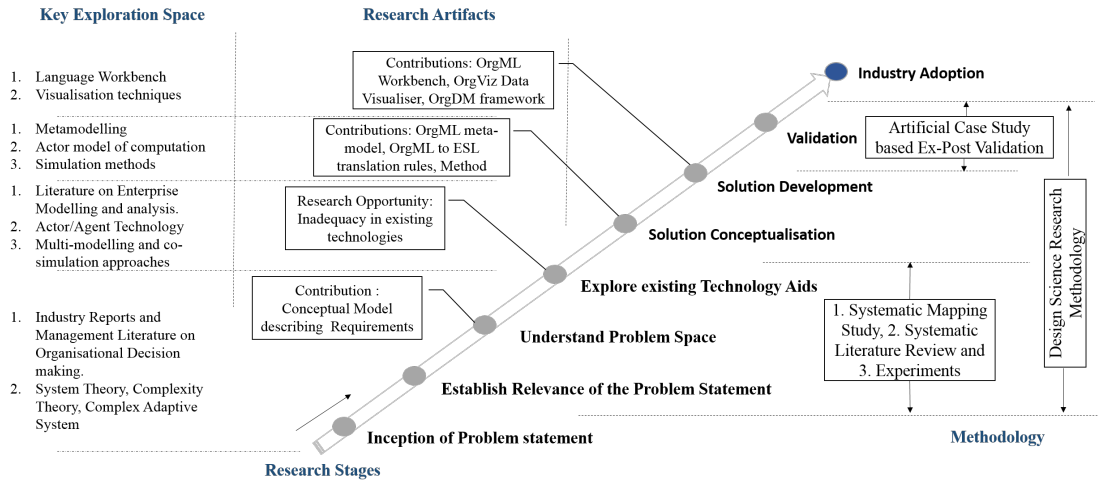


Figure 8.1 Research Overview – phases, methodology, artifacts and exploration space

The solution space for these problems raises further issues in that there are several exploration paths along inferential approaches, Operational Research (OR) approaches, top-down and bottom-up simulation approaches. Hence, the review on existing multi-modelling and co-simulation approaches and actor/agent technologies identified further requirements.

This research proposes a new actor-based simulation aid to approach organisational decision making in a systematic manner. The inadequacies identified by systematic literature reviews are addressed by three novel research contributions that include – (i) OrgML meta-model and specification language to capture necessary information for organisation decision-making in an intuitive and domain specific manner, (ii) an analysis approach that uses OrgML as specification, ESL as underlying simulation technology and OrgViz Data Visualiser as a visualisation aid for required *what-if* analyses, and (iii) a method to capture required information, ascertain model/specification validity and perform iterative *what-if* analyses. Figure 8.1 provides a summary of methodology, research artifacts and the direction of research undertaken in this thesis.

This concluding chapter highlights the research contributions made in this research, discusses limitations and future work of this thesis. The research contributions and their significance are discussed in section 8.1. The limitations and future work of this thesis are briefly revisited in section 8.2 and section 8.4. The thesis is concluded with a concluding remark in section 8.5.

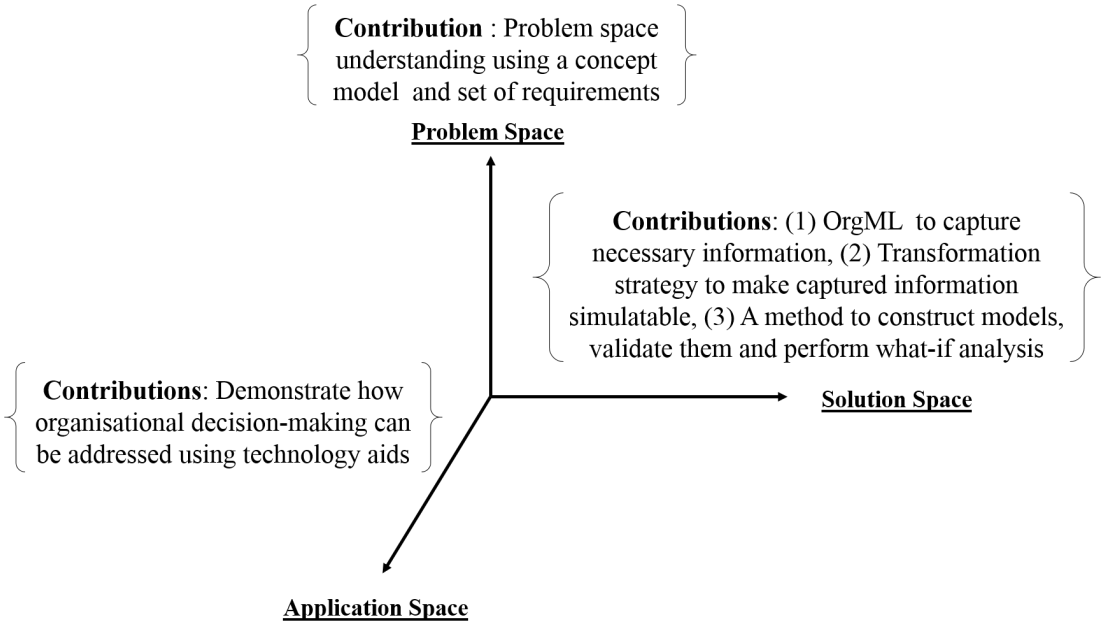


Figure 8.2 Dimensions of research contributions

8.1 Research contributions and significance

Research contributions of this thesis span across three dimensions that include: (i) problem space understanding, (ii) technology *improvements* for solutions space, and (iii) an effective application of an actor-based simulation technology in organisational decision-making as shown in Figure 8.2.

Problem understanding

The research questions pertaining to what information is necessary and what are the characteristics of the information for effective organisational decision making lead to a precise understanding of the problem space. The relevant aspects and concepts of organisational decision-making are discerned from the management literature and expressed using a meta-model as presented in Figure 3.6 of Chapter 3. The meta-model primarily captures *structure*, *behaviour*, *state*, *trace* and *environment* aspects of an organisation. It also consider three decision-making concepts that include: *goal*, *measure* and *lever*. In addition, a set of *requirements* that exhibit the characteristics of complex organisations are identified and enumerated in tabular form (as presented in Table 3.3). The characteristics are: *modularity*, *composability*, *reactiveness*, *autonomy*, *intentionality*, *adaptiveness*, *uncertainty* and *temporal behaviour*.

This research argues that the proposed meta-model representing organisational decision-making aspects/concepts and associated requirements collectively characterises the problem space. In this research, these artifacts are used as the reference for literature review, development of research contributions and research validation. Potentially these artifacts can also be used in other organisational decision-making related research as they are validated using near real-life case studies and received wide acceptance from scholastic communities (as conference/journal publications, such as RP [4], RP [6], RP [9] and RP [11] – *see* publication section).

Solution space

Research questions that focus on how to capture necessary information, what kind of analysis is needed and how the analyses can be performed for an effective organisational decision-making leads to three research contributions:

- **OrgML meta-model and specification language**: The proposed OrgML meta-model (and OrgML specification language) can capture necessary information of an organisational decision-making in an intuitive and domain specific manner. The proposed meta-model supports the specification that can sufficiently capture necessary organisational aspects, decision-making concepts and characteristics as defined as problem space understanding. The OrgML meta-model is designed such a way that a complex organisation can be used to faithfully represent a real-world complex organisation. Therefore, the use of OrgML is not just limited to organisational decision-making but any business problem which needs precise understanding of an organisation may use this specification.

- **An approach to make captured information amenable for simulation**: Proposed approach to transform OrgML specification into the ESL specification using a model-to-model transformation technique help decision makers to perform desired *what-if* analysis.

- **Method**: An integrated and iterative method to construct a purposive simulation model leading to organisational decision-making is proposed. It is a generic method – therefore it can be used with other models and analysis techniques. For an instance, this method is used to explore a multi-modelling and co-simulation approach that uses i* , BPMN, and Stock-and-Flow as discussed in Appendix F and reported in [21] (*i.e.* RP [1] in research publication list).

Application space

The proposed approach and research artifacts are applied to decision space exploration of a range of organisational decision-making problems, such as *Management science*, *Carnegie model*, *Incremental process model* and *Garbage Can model* as discussed in Chapter 7.

In addition to the organisation decision-making space discussed in this thesis, the proposed approach can be leveraged to design space exploration leading to enterprise transformation initiative [169] and construct a *digital twin* [45] of business organisations. For instance, the proposed approach can be adopted to explore design space of a to-be system. The design alternatives can be captured using separate OrgML specifications that can be evaluated through simulation to identify which design option is the best among available alternatives to define a to-be organisation in an enterprise transformation initiative. Similarly, the OrgML model/specification can be used for creating a *digital twin* of a business organisation as the OrgML is capable of mimicking the real organisation, its units and sub-units as they exist in reality.

8.2 Limitations

The limitations identified while evaluating the proposed research artifacts can be broadly classified into three categories – specification related limitations, implementation related limitations and scoping limitation. The specification limitations include the inadequacy to express spatial relationship [214], absolute time and continuous time. Implementation related limitations are mainly due to a partial proof-of-concept implementation of the proposed research artifacts as opposed to the production quality. The key implementation limitations are: limited graph types in the OrgViz Data visualiser, partial automation of OrgML-to-ESL transformation and the lack of industry scale implementation of the presented approach and conceptualised framework.

The other limitations of the proposed approach is due to the scope of this research. The socio-political aspects of the decision making are not considered, and use of the proposed approach in a real context is not considered as they are out of scope of this thesis.

8.3 Reflection

While applying research contributions on different case studies, it is realised that the proposed actor-based modelling abstraction can be utilised in a wider business context as it has a capability to closely imitate complex business systems. For an example, the proposed modelling abstraction

can be effectively used as a foundational basis for various *digital twin* [45] initiatives. The use of bottom-up simulation technique further improves its utilisation as it helps to understand the emergent behaviour of the complex systems and uncertain environments where they operate.

Reflecting on the experiences of developing multiple case studies in research validation phase, it is realised that this research chiefly focuses on technical *efficacy* of the research contributions over the state-of-the-art modelling and analysis technologies. While *efficacy* is an important validation criteria as suggested in Software Engineering and Information System research [203, 160], the need for evaluating *Environmental* and *Structural* factors [160], such as *Ease of use* and *Simplicity* (*see* Figure 2.2), are also found to be critical to effectively utilise a technical solution in a business problem (*e.g.*, organisational decision-making). Development of intuitive graphical notations or hybrid (graphics + text) symbols to achieve *cognitive effectiveness* while representing complex systems as described by Daniel Moody in [142] is a useful research direction to make this research more amenable to the business stakeholders.

From the methodological perspective, this thesis refines the canonical DSR methodology using SMS, SLR and experimentations. I found this refinement is useful for addressing critical business problems where the problem space and solution space are not well-defined (such as orgnisational decision-making and enterprise transformation). The SMS and SLR collectively help to understand the problem space and solution space from the perspective of the existing knowledge base. The SMS covers the breadth of a (problem or solution) domain, whereas the SLR explores the depth of a domain. The experimentations with the existing tools from the solution space help to identify precise gaps and establish the relevance of a research problem. The use of multiple case studies as opposed to a single comprehensive case study is another useful extension to the methodology. Multiple case studies focusing on different business cases is a practical consideration for improving the rigour of the proposed solution and research contributions as it is a difficult proposition to conceptualise (or identify) one comprehensive case study that closely represents the reality and includes all validation scenarios.

8.4 Future research directions

This research has opened the door for research opportunities along two major dimensions – technology development for organisational decision-making and application of the technology aids in a range of organisational business problems. Foremost among the future work along technology development is to provide more technology aids for decision makers. The technology

aids to compute the distance of an organisation from its current state to a desired state, and identify the 'next' possible course of action/lever that should be considered for decision space exploration are useful contributions. The graphical representation of the OrgML specification (as indicated in Appendix B) and an industry scale implementation of the proposed approach are definite technology development activities. The other significant technology enhancement activities are – (i) supporting multi-modelling specification in OrgML, (ii) support for co-simulation to combine Enterprise Simulation Language (ESL), Stock and Flow (SnF) and Operational Research (OR) techniques, and (iii) develop the necessary framework to combine inferential technique and simulation technique for organisational decision making as discussed in Chapter 7.

On the other hand, this thesis has demonstrated how a technology aid can be used in organisational decision-making. The use of the proposed technology aid (along with the extensions) in exploring the design space for an enterprise transformation, creating a suitable environment to experiment with game theoretic aspects of business environment [18], and realisation of the digital-twin [45] of the business organisations in business 4.0 initiative[1] are areas for future application of this work.

8.5 Concluding remark

The aim of this research was to conceptualise and develop a suitable technology aid to approach organisational decision-making using quantitative what-if analysis as opposed to human intuitions. The exploration of the problem space from management lens bought forth the necessary information for an effective organisational decision-making. The deeper analysis of the management and IS literature reestablishes the need for all six interrogative aspects: *why*, *what*, *how*, *when*, *where*, and *who* as recommended in Zachman framework. The analysis also shows the importance of socio-technical characteristics, such as *modularity*, *compositional*, *reactive*, *autonomous*, *intentional*, *adaptive*, *uncertainty* and *temporal behaviour*, to comprehend modern organisation.

The state-of-the-art modelling and analysis techniques that include enterprise modelling and analysis techniques and actor/agent languages are found to be inadequate for the quantitative analysis of the modern organisation. This research has developed a new and novel approach to model complex organisations and quantitatively analyse various courses of action using the

[1] sites.tcs.com/insights/perspectives/category/business-4-0

constructed model. While conceptualising, developing and validating, several hypotheses about the expected technology aids for organisational decision-making are validated. For example, the relevance of an actor-based bottom-up simulation approach to understand complex organisation is established. The efficacy of OrgML based approach that advocates six interrogative aspects, an extended form of *actor*, richer composability, uncertain and temporal behaviour, and a machine-interpretable specification to overcome the limitations of the state-of-art and practice of modelling and analysis techniques is demonstrated. The importance of an integrated approach to utilise technology aids for model creation, model validation and perform *what-if* analyses are demonstrated. How an approach that combines simulation research and management view of decision making can serve the needs of quantitative, evidence-driven, informed organganisational decision-making is also shown in this thesis.

The OrgML based modelling abstraction developed through this research establishes a pragmatic conceptual modelling framework for a wide range of organisational decision-making problems. The presented actor-based simulation approach and supported technology aid collectively improve the state-of-the-art *what-if* analysis techniques, by utilising a proposed method that combines the theory and practice of organisational decision-making. The proposed method also demonstrates how technological aids can be effectively utilised in a variety of complex organisational decision-making problems.

Bibliography

[1] Adebesin, F., Kotzé, P., and Gelderblom, H. (2011). Design research as a framework to evaluate the usability and accessibility of the digital doorway. In *Design, Development Research Conference, Cape Peninsula University of Technology*, pages 306–323. CSIR.

[2] Agha, G. (1986a). *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, USA.

[3] Agha, G. (1986b). An Overview of Actor Languages. *SIGPLAN Not.*, 21(10):58–67.

[4] Aigner, W., Miksch, S., Müller, W., Schumann, H., and Tominski, C. (2007). Visualizing time-oriented data – a systematic view. *Computers & Graphics*, 31(3):401–409.

[5] Allen, J. (2013). *Effective Akka*. O'Reilly Media, Inc.

[6] Altbach, P. (2007). The Dilemmas of Ranking in: International Higher Education. *Journal of Studies in International Education*, 3(6):121–133.

[7] Amagoh, F. (2008). Perspectives on organizational change: systems and complexity theories. *The Innovation Journal: The public sector innovation journal*, 13(3):1–14.

[8] Anderson, D., Sweeney, D., Williams, T., Camm, J., and Cochran, J. (2015). *An introduction to management science: quantitative approaches to decision making*. Cengage Learning.

[9] Anderson, M. J. (2001). A new method for non-parametric multivariate analysis of variance. *Austral ecology*, 26(1):32–46.

[10] Anderson, P. (1999). Perspective: Complexity theory and organization science. *Organization science*, 10(3):216–232.

[11] Arcangeli, S. R. J.-P., Migeon, F., and Rougemaille, S. (2008). Javact: a java middleware for mobile adaptive agents. *Lab. IRIT, University of Toulouse*.

[12] Armstrong, J. (1996). Erlang - a Survey of the Language and its Industrial Applications. In *In Proceedings of the symposium on industrial applications of Prolog (INAP)*, page 8.

[13] Armstrong, J., Virding, R., Wikström, C., and Williams, M. (1993). *Concurrent programming in ERLANG*. Prentice Hall.

[14] Ashby, W. R. (1981). *Mechanisms of Intelligence: Ashbys Writings on Cybernetics*. Eipiphiny Society.

[15] Astley, M. (1998). The actor foundry: A java-based actor programming environment. *University of Illinois at Urbana-Champaign: Open Systems Laboratory*.

[16] Ayres, J. and Eisenbach, S. (2009). Stage: Python with actors. In *Proceedings of the 2009 ICSE Workshop on Multicore Software Engineering*, pages 25–32. IEEE Computer Society.

[17] Bailey, I. (2008). Brief introduction to MODAF with v1. 2 updates. *IET Seminar on Enterprise Architecture Framework, London*, pages 1–18.

[18] Bandyopadhyay, S. and Pathak, P. (2007). Knowledge sharing and cooperation in outsourcing projects – A game theoretic analysis. *Decision support systems*, 43(2):349–358.

[19] Barat, S. (2016). A simulation based aid for complex dynamic decision making. In *PoEM Doctoral Consortium*, pages 22–31.

[20] Barat, S. (2017). An actor-based bottom-up Simulation aid for complex dynamic decision making. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 275–278. Springer.

[21] Barat, S., Kulkarni, V., and Barn, B. (2018a). Towards Improved Organisational Decision-Making–A Method and Tool-chain. *Enterprise Modelling and Information Systems Architectures*, 13:6–31.

[22] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2016a). A conceptual model for organisational decision-making and its possible Realisations. In *ESM 2016: 30th Annual European Simulation and Modelling Conference*, pages 174–176.

[23] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2016b). A simulation-based aid for organisational decision-making. In *ICSOFT-EA 2016: 11th International Conference on Software Engineering and Applications*, pages 109–116.

[24] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2016c). Enterprise modeling as a decision making aid: A systematic mapping study. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 289–298. Springer.

[25] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2017a). A domain specific language for complex dynamic decision making. In *ESM 2017: 31st Annual European Simulation and Modelling Conference*, pages 135–142.

[26] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2017b). A method for effective use of enterprise modelling techniques in complex dynamic decision making. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 319–330. Springer.

[27] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2017c). A Model based Realisation of Actor Model to Conceptualise an Aid for Complex Dynamic Decision-making. In *MODELSWARD 2017*, pages 605–616.

[28] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2017d). An actor-model based bottom-up simulation—An experiment on Indian demonetisation initiative. In *Winter Simulation Conference (WSC), 2017*, pages 860–871. IEEE.

[29] Barat, S., Kulkarni, V., Clark, T., and Barn, B. (2018b). A Model Based Approach for Complex Dynamic Decision-Making. In *Communications in Computer and Information Science*, volume 880, pages 94–118. Springer.

[30] Barat, S., Rajbhoj, A., Kumar, P., and Kulkarni, V. (2017e). A Case Study Exploring Suitability of Bottom Up Modelling and Actor-based Simulation for Decision Making. In *Proceedings of Modeling Symposium*, pages 1–6.

[31] Barjis, J. (2008). Enterprise, organization, modeling, simulation: Putting pieces together. *CEUR Workshop Proceedings*, 338:1–5.

[32] Barros, T., Ameur-Boulifa, R., Cansado, A., Henrio, L., and Madelaine, E. (2009). Behavioural models for distributed Fractal components. *annals of telecommunications-annales des télécommunications*, 64(1-2):25–43.

[33] Barua, N., Choudhury, M., and Borkakoty, A. (2009). *Business process outsourcing*. Daya Publishing House.

[34] Beckermann, A., Flohr, H., and Kim, J. (1992). *Emergence or reduction?: Essays on the prospects of nonreductive physicalism*. Walter de Gruyter.

[35] Bell, D. E., Raiffa, H., and Tversky, A. (1988). *Decision making: Descriptive, normative, and prescriptive interactions*. Cambridge University Press.

[36] Bellifemine, F., Poggi, A., and Rimassa, G. (1999). JADE–A FIPA-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London.

[37] Bernus, P., Mertins, K., and Schmidt, G. J. (2013). *Handbook on architectures of information systems*. Springer Science & Business Media.

[38] Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

[39] Bettini, L. (2016). *Implementing domain-specific languages with Xtext and Xtend*. Packt Publishing Ltd.

[40] Bock, A., Frank, U., Bergmann, A., and Strecker, S. (2016). Towards Support for Strategic Decision Processes Using Enterprise Models: A Critical Reconstruction of Strategy Analysis Tools. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 41–56. Springer.

[41] Boehm, B. W. (1984). Software engineering economics. *IEEE transactions on Software Engineering*, (1):4–21.

[42] Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287.

[43] Boon, M. and Knuuttila, T. (2009). Models as epistemic tools in engineering sciences: A pragmatic approach. *Handbook of the philosophy of technological sciences*, 9:687–719.

[44] Borshchev, A. (2013). *The big book of simulation modeling: multimethod modeling with AnyLogic 6*. AnyLogic North America Chicago.

[45] Boschert, S. and Rosen, R. (2016). Digital twin – the simulation aspect. In *Mechatronic Futures*, pages 59–74. Springer.

[46] Briot, J.-P. (1999). Actalk: A framework for object-oriented concurrent programming-design and experience. *Object-Oriented Parallel and Distributed Programming*, pages 209–231.

[47] Camus, B., Bourjot, C., and Chevrier, V. (2015). Combining DEVS with multi-agent concepts to design and simulate multi-models of complex systems (WIP). In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pages 85–90.

[48] Candes, E. J. and Tao, T. (2005). Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215.

[49] Casti, J. L. (1994). *Complexification explaining a paradoxical world through the science of surprise*. HarperPerennial - A Division of Harper Collins Publishers.

[50] Chen, D., Daclin, N., et al. (2006). Framework for enterprise interoperability. In *Proc. of IFAC Workshop EI2N*, pages 77–88. Bordeaux.

[51] Chen, D., Vallespir, B., and Doumeingts, G. (1997). GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology. *Computers in industry*, 33(2-3):387–394.

[52] Clark, T., Barn, B., Kulkarni, V., and Barat, S. (2017a). Querying histories of organisation simulations. *Information Systems Development (ISD) - Advances in Methods, Tools and Management*, 9:1–12.

[53] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2017b). Actor monitors for adaptive behaviour. In *Proceedings of the 10th Innovations in Software Engineering Conference*, pages 85–95. ACM.

[54] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2017c). ESL: An Actor-Based Platform for Developing Emergent Behaviour Organisation Simulations. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 311–315. Springer.

[55] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2017d). Generating Filmstrip Models from Actor-Based Systems. In *MODELS (Satellite Events) 2017*, pages 576–582.

[56] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2017e). The Construction and Interrogation of Actor Based Simulation Histories. In *ER Forum/Demos 2017*, pages 320–333.

[57] Clark, T., Kulkarni, V., Barat, S., and Barn, B. (2018). A homogeneous actor-based monitor language for adaptive behaviour. In *Programming with Actors*, pages 216–244. Springer.

[58] Clark, T., Kulkarni, V., Barn, B., France, R., Frank, U., and Turk, D. (2014). Towards the model driven organization. In *47th Hawaii International Conference on System Sciences (HICSS)*, pages 4817–4826. IEEE.

[59] Clebsch, S. (2015). The pony programming language. The Pony Developers. http://www.ponylang.org/.

[60] Cohen, M. D., March, J. G., and Olsen, J. P. (1972). A garbage can model of organizational choice. *Administrative science quarterly*, pages 1–25.

[61] Collier, N. (2003). Repast: An extensible framework for agent simulation. *The University of Chicago's Social Science Research*, 36:2003.

[62] Conrath, D. W. (1967). Organizational decision making behavior under varying conditions of uncertainty. *Management Science*, 13(8):B–487.

[63] Conway, J. (1970). The game of life. *Scientific American*, 223(4):4.

[64] Cook, T. D. and Campbell, D. T. (1979). *Quasi-experimentation: Design and analysis for field settings*, volume 3. Rand McNally Chicago.

[65] Currall, S. C. and Towler, A. J. (2003). Research methods in management and organizational research: Toward integration of qualitative and quantitative techniques. Sage Publications.

[66] Cyert, R. M., March, J. G., et al. (1963). A behavioral theory of the firm. *Englewood Cliffs, NJ*, 2.

[67] Cyert, R. M., Simon, H. A., and Trow, D. B. (1956). Observation of a business decision. *The Journal of Business*, 29(4):237–248.

[68] Daft, R. (2012). *Organization theory and design*. Nelson Education.

[69] Daft, R. L. and Lewin, A. Y. (1990). Can organization studies begin to break out of the normal science straitjacket? An editorial essay. *Organization Science*, 1(1):1–9.

[70] Daft, R. L. and Marcic, D. (2016). *Understanding management*. Nelson Education.

[71] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[72] Dedecker, J., Van Cutsem, T., Mostinckx, S., D'Hondt, T., and De Meuter, W. (2006). Ambient-oriented programming in ambienttalk. In *European Conference on Object-Oriented Programming*, pages 230–254. Springer.

[73] Dietz, J. L. and Hoogervorst, J. A. (2008). Enterprise ontology in enterprise engineering. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 572–579. ACM.

[74] Drazin, R. and Sandelands, L. (1992). Autogenesis: A perspective on the process of organizing. *Organization Science*, 3(2):230–249.

[75] Eisenhardt, K. M. and Zbaracki, M. J. (1992). Strategic decision making. *Strategic management journal*, 13(S2):17–37.

[76] Erdweg, S., Van Der Storm, T., Völter, M., Boersma, M., Bosman, R., Cook, W. R., Gerritsen, A., Hulshout, A., Kelly, S., Loh, A., et al. (2013). The state of the art in language workbenches. In *International Conference on Software Language Engineering*, pages 197–217. Springer.

[77] Euzenat, J. (2001). Towards a principled approach to semantic interoperability. In *Proc. IJCAI 2001 workshop on ontology and information sharing*, pages 19–25.

[78] Forrester, J. W. (1994). System dynamics, systems thinking, and soft OR. *System dynamics review*, 10(2-3):245–256.

[79] Fowler, M. (2010). *Domain-specific languages*. Pearson Education.

[80] Frank, U. (2002). Multi-perspective Enterprise Modeling (MEMO) conceptual framework and modeling languages. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 1258–1267. IEEE.

[81] Frank, U., Squazzoni, F., and Troitzsch, K. G. (2009). EPOS-Epistemological perspectives on simulation: An introduction. In *Epistemological Aspects of Computer Simulation in the Social Sciences*, pages 1–11. Springer.

[82] Goldsman, D., Nance, R. E., and Wilson, J. R. (2010). A brief history of simulation revisited. In *Proceedings of the winter simulation conference*, pages 567–574.

[83] Goralwalla, I. A., Özsu, M. T., and Szafron, D. (1998). An object-oriented framework for temporal data models. In *Temporal Databases: Research and Practice*, pages 1–35. Springer.

[84] Grangel, R., Chalmeta, R., and Campos, C. (2007). Using UML profiles for enterprise knowledge modelling. In *EDOC Conference Workshop, 2007. EDOC'07. Eleventh International IEEE*, pages 125–132. IEEE.

[85] Gregor, S. (2006). The nature of theory in information systems. *MIS quarterly*, pages 611–642.

[86] Gregor, S. and Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS quarterly*, 37(2).

[87] Gregor, S. and Jones, D. (2007). The anatomy of a design theory. *Journal of the Association for Information Systems*, 8(5):312.

[88] Grignard, A., Taillandier, P., Gaudou, B., Vo, D. A., Huynh, N. Q., and Drogoul, A. (2013). GAMA 1.6: Advancing the art of complex agent-based modeling and simulation. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 117–131. Springer.

[89] Guba, E. G. and Lincoln, Y. S. (1982). Epistemological and methodological bases of naturalistic inquiry. *Educational Technology Research and Development*, 30(4):233–252.

[90] Haller, P. and Odersky, M. (2009). Scala actors: Unifying thread-based and event-based programming. *Theoretical Computer Science*, 410(2):202–220.

[91] Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3):231–274.

[92] Harnett, T. (2011). Consensus-Oriented Decision-Making. *Gabriola Island, BC: Canada New Society Publishers*.

[93] Harrison, R. (2007). *TOGAF version 8.11 enterprise edition*. Van Haren Publishing.

[94] Hevner, A. and Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems*, pages 9–22. Springer.

[95] Hevner, A., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

[96] Hewitt, C. (2010). Actor model of computation: scalable robust information systems. *arXiv preprint arXiv:1008.1459*.

[97] Hewitt, C. and Smith, B. (1975). A plasma primer. *Draft. Cambridge, Massachusetts: MIT Artificial Intelligence Laboratory*.

[98] Hindriks, K. V., De Boer, F. S., Van der Hoek, W., and Meyer, J.-J. C. (1999). Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401.

[99] Holland, J. H. (2006). Studying complex adaptive systems. *Journal of Systems Science and Complexity*, 19(1):1–8.

[100] Iacob, M., Jonkers, D. H., Lankhorst, M., Proper, E., and Quartel, D. D. (2012). ArchiMate 2.0 Specification: The Open Group. Van Haren Publishing.

[101] IFIP-IFAC Task Force on Architectures for Enterprise Integration (2003). GERAM: The Generalised Enterprise Reference Architecture and Methodology: Vesion 1.6. 3 (Final). *Handbook on enterprise architecture*, pages 21–63.

[102] Iivari, J. (2007). A paradigmatic analysis of information systems as a design science. *Scandinavian journal of information systems*, 19(2):5.

[103] Jang, M.-W. (2004). The actor architecture manual. *Department of Computer Science, University of Illinois at Urbana-Champaign*.

[104] Kafura, D. G. and Lee, K. H. (1989). *ACT++: building a concurrent C++ with actors*. Virginia Polytechnic Institute & State University.

[105] Kahneman, D., Lovallo, D., and Sibony, O. (2011). Before you make that big decision. *Harvard business review*, 89(6):50–60.

[106] Kaplan, B. and Duchon, D. (1988). Combining qualitative and quantitative methods in information systems research: a case study. *MIS quarterly*, pages 571–586.

[107] Keller, W. (1997). Mapping objects to tables. In *Proc. of European Conference on Pattern Languages of Programming and Computing, Kloster Irsee, Germany*, volume 206, page 207. Citeseer.

[108] Kickert, W. J. (1980). *Organisation of decision-making: a systems-theoretical approach*. North-Holland Publishing Company.

[109] Kim, W. (1997). *ThAL: An actor system for efficient and scalable concurrent computing*. PhD thesis, University of Illinois at Urbana-Champaign.

[110] Kinny, D., Georgeff, M., and Rao, A. (1996). A methodology and modelling technique for systems of BDI agents. *Agents breaking away*, pages 56–71.

[111] Kitchenham, B. and Brereton, P. (2013). A systematic review of systematic review process research in software engineering. *Information and software technology*, 55(12):2049–2075.

[112] Kleppe, A. G., Warmer, J., Warmer, J. B., and Bast, W. (2003). *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional.

[113] Kothari, C. R. (2004). *Research methodology: Methods and techniques*. New Age International.

[114] Kotusev, S., Singh, M., and Storey, I. (2015). Investigating the Usage of Enterprise Architecture Artifacts. In *ECIS, Research-in-Progress Papers. Paper 15*.

[115] Krogstie, J. (2008). Using EEML for combined goal and process oriented modeling: A case study. *CEUR Workshop Proceedings*, 337:112–129.

[116] Krogstie, J., Lindland, O. I., and Sindre, G. (1995). Defining quality aspects for conceptual models. In *Information System Concepts*, pages 216–231. Springer.

[117] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2014). Model Based Enterprise Simulation and Analysis - A Pragmatic Approach Reducing the Burden on Experts. In *ER Workshops 2014*, pages 3–12.

[118] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2015a). A wide-spectrum approach to modelling and analysis of organisation for machine-assisted decision-making. In *Workshop on Enterprise and Organizational Modeling and Simulation*, pages 87–101.

[119] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2015b). Supporting organisational decision making in presence of uncertainty. In *European Modeling and Simulation Symposium, EMSS 2017*, pages 87–101.

[120] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2015c). Toward overcoming accidental complexity in organisational decision-making. In *Model Driven Engineering Languages and Systems (MODELS)*, pages 368–377.

[121] Kulkarni, V., Barat, S., Clark, T., and Barn, B. (2015d). Using simulation to address intrinsic complexity in multi-modelling of enterprises for decision making. In *Proceedings of the Conference on Summer Computer Simulation*, pages 1–11. Society for Computer Simulation International.

[122] Kulkarni, V., Barat, S., and Roychoudhury, S. (2012). Towards business application product lines. In *International Conference on Model Driven Engineering Languages and Systems*, pages 285–301. Springer.

[123] Langley, A., Mintzberg, H., Pitcher, P., Posada, E., and Saint-Macary, J. (1995). Opening up decision making: The view from the black stool. *organization Science*, 6(3):260–279.

[124] Levitt, B. and March, J. G. (1988). Organizational learning. *Annual review of sociology*, 14(1):319–338.

[125] Lieberman, H. (1981). A preview of ACT 1. *MIT Artificial Intelligence Laboratory*, A.I. Memo No. 625.

[126] Locke, E. (2011). *Handbook of principles of organizational behavior: Indispensable knowledge for evidence-based management*. John Wiley & Sons.

[127] Loucopoulos, P., Stratigaki, C., Danesh, M. H., Bravos, G., Anagnostopoulos, D., and Dimitrakopoulos, G. (2015). Enterprise capability modeling: concepts, method, and application. In *Enterprise Systems (ES), 2015 International Conference on*, pages 66–77. IEEE.

[128] Lukman, R., Krajnc, D., and Glavič, P. (2010). University ranking using research, educational and environmental indicators. *Journal of Cleaner Production*, 18(7):619–628.

[129] Macal, C. M. and North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of simulation*, 4(3):151–162.

[130] March, J. G. (1994). *Primer on decision making: How decisions happen*. Simon and Schuster.

[131] March, S. T. and Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, 15(4):251–266.

[132] McDermott, T., Rouse, W., Goodman, S., and Loper, M. (2013). Multi-level modeling of complex socio-technical systems. *Procedia Computer Science*, 16:1132–1141.

[133] Mcmillan, C. J. (1980). Qualitative models of organisational decision-making. *Journal of General Management*, 5(4):22–39.

[134] Meadows, D. H. and Wright, D. (2008). *Thinking in systems: A primer*. Chelsea Green Publishing.

[135] Meissner, P., Sibony, O., and Wulf, T. (2015). Are you ready to decide? *McKinsey Quarterly, April*, 8.

[136] Mendling, J. (2008). Event-driven Process Chains (EPC). In *Metrics for Process Models*, pages 17–57. Springer.

[137] Menzel, C. and Mayer, R. J. (1998). The IDEF family of languages. In *Handbook on architectures of information systems*, pages 209–241. Springer.

[138] Michalski, R. S. (1993). Inferential theory of learning as a conceptual basis for multistrategy learning. *Machine Learning*, 11(2-3):111–151.

[139] Michelson, B. M. (2006). Event-driven architecture overview. *Patricia Seybold Group*, 2.

[140] Miller, M. S., Tribble, E. D., and Shapiro, J. (2005). Concurrency among strangers. In *International Symposium on Trustworthy Global Computing*, pages 195–229. Springer.

[141] Mintzberg, H., Raisinghani, D., and Theoret, A. (1976). The structure of unstructured decision processes. *Administrative science quarterly*, pages 246–275.

[142] Moody, D. (2009). The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779.

[143] Moody, D., Sindre, G., Brasethvik, T., and Sølvberg, A. (2003). Evaluating the quality of information models: empirical testing of a conceptual model quality framework. In *Proceedings of the 25th international conference on software engineering*, pages 295–305. IEEE Computer Society.

[144] Mooney, C. Z. (1997). *Monte carlo simulation*, volume 116. Sage Publications.

[145] Myers, M. D. et al. (1997). Qualitative research in information systems. *Management Information Systems Quarterly*, 21(2):241–242.

[146] Narayanan, K. and Ramaswamy, S. (2005). Specifications for mapping UML models to XML schemas. In *Proceedings of the 4th Workshop in Software Model Engineering (WiSME 2005)*, pages 1–10.

[147] Nelson, H. J., Poels, G., Genero, M., and Piattini, M. (2012). A conceptual modeling quality framework. *Software Quality Journal*, 20(1):201–228.

[148] Nowak, A., Vallacher, R. R., Tesser, A., and Borkowski, W. (2000). Society of self: The emergence of collective properties in self-structure. *Psychological review*, 107(1):39.

[149] Oates, B. J. (2005). *Researching information systems and computing*. Sage.

[150] O'Connor, T. and Wong, H. Y. (2002). Emergent properties. *stanford.library.sydney.edu.au*.

[151] OMG (2004). 2.0 Superstructure Specification. *OMG, Needham*.

[152] Orlikowski, W. J. and Baroudi, J. J. (1991). Studying information technology in organizations: Research approaches and assumptions. *Information systems research*, 2(1):1–28.

[153] Panetto, H. (2003). UML semantics representation of enterprise modelling constructs. In *Enterprise Inter-and Intra-Organizational Integration*, pages 381–387. Springer.

[154] Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77.

[155] Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *12th international conference on evaluation and assessment in software engineering*, volume 17, pages 1–10.

[156] Peterson, J. L. (1981). Petri Net Theory and the Modeling of Systems. *PrenticeHall, inc, Englewood Cliffst1*, 981.

[157] Pettigrew, A. M. (2014). *The politics of organizational decision-making*. Routledge.

[158] Pnueli, A. (1977). The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57. IEEE.

[159] Pokahr, A., Braubach, L., and Lamersdorf, W. (2005). Jadex: A BDI reasoning engine. In *Multi-agent programming*, pages 149–174. Springer.

[160] Prat, N., Comyn-Wattiau, I., and Akoka, J. (2014). Artifact Evaluation in Information Systems Design-Science Research-a Holistic View. In *PACIS*, page 23. Citeseer.

[161] Pries-Heje, J., Baskerville, R., and Venable, J. (2008). Strategies for design science research evaluation. *ECIS 2008 proceedings*, pages 1–12.

[162] Rao, A. S. (1996). AgentSpeak (L): BDI agents speak out in a logical computable language. In *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 42–55. Springer.

[163] Rao, A. S., Georgeff, M. P., et al. (1995). BDI agents: from theory to practice. In *ICMAS*, volume 95, pages 312–319.

[164] Raymond, K. (1995). Reference model of open distributed processing (RM-ODP): Introduction. In *Open distributed processing*, pages 3–14. Springer.

[165] Rettig, M. (2010). retlang: Message based concurrency in .NET. https://www.findbestopensource.com/product/retlang.

[166] Robinson, S. (2008). Conceptual modelling for simulation Part I: definition and requirements. *Journal of the operational research society*, 59(3):278–290.

[167] Rodriguez, S., Gaud, N., and Galland, S. (2014). SARL: a general-purpose agent-oriented programming language. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 103–110. IEEE.

[168] Rolland, C., Nurcan, S., and Grosz, G. (1999). Enterprise knowledge development: the process view. *Information & management*, 36(3):165–184.

[169] Rouse, W. B. and Baba, M. L. (2006). Enterprise transformation. *Communications of the ACM*, 49(7):66–72.

[170] Rumsfeld, D. (2011). *Known and unknown: A memoir*. Penguin.

[171] Saltelli, A., Chan, K., Scott, E. M., et al. (2000). *Sensitivity analysis*, volume 1. Wiley New York.

[172] Sandkuhl, K., Fill, H.-G., Hoppenbrouwers, S., Krogstie, J., Leue, A., Matthes, F., Opdahl, A. L., Schwabe, G., Uludag, Ö., and Winter, R. (2016). Enterprise modelling for the masses – from elitist discipline to common practice. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 225–240. Springer.

[173] Sandkuhl, K., Stirna, J., Persson, A., and Wißotzki, M. (2014). Enterprise modeling. *Tackling Business Challenges with the 4EM Method. Springer*, 309.

[174] Sargent, R. G. (2005). Verification and validation of simulation models. In *Proceedings of the 37th conference on Winter simulation*, pages 130–143.

[175] Scheer, A.-W. and Nüttgens, M. (2000). ARIS architecture and reference models for business process management. In *Business process management*, pages 376–389. Springer.

[176] Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons.

[177] Shapira, Z. (2002). *Organizational decision making*. Cambridge University Press.

[178] Siebert, J., Ciarletta, L., and Chevrier, V. (2010). Agents and artefacts for multiple models co-evolution: building complex system simulation as a set of interacting models. In *9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 509–516. International Foundation for Autonomous Agents and Multiagent Systems.

[179] Silva, S. F. and Catarci, T. (2000). Visualization of linear time-oriented data: A survey. In *Web Information Systems Engineering, 2000. Proceedings of the First International Conference on*, volume 1, pages 310–319. IEEE.

[180] Simon, H. A. (1959). Theories of decision-making in economics and behavioral science. *The American economic review*, 49(3):253–283.

[181] Simon, H. A. (1977). *The New Science of Management Decision*. Prentice Hall PTR.

[182] Simon, H. A. (1979). Rational decision making in business organizations. *The American economic review*, 69(4):493–513.

[183] Simon, H. A. (1991). The architecture of complexity. In *Facets of systems science*, pages 457–476. Springer.

[184] Simon, H. A. (1996). *The sciences of the artificial*. MIT press.

[185] Sipp, C. M. and Elias, C. (2012). *Real Options and Strategic Technology Venturing: A New Paradigm in Decision Making*, volume 31. Springer Science & Business Media.

[186] Sirjani, M., Movaghar, A., Shali, A., and De Boer, F. S. (2004). Modeling and verification of reactive systems using Rebeca. *Fundamenta Informaticae*, 63(4):385–410.

[187] Srinivasan, S. and Mycroft, A. (2008). Kilim: Isolation-typed actors for java. In *European Conference on Object-Oriented Programming*, pages 104–128. Springer.

[188] Sturman, D. C. and Agha, G. A. (1994). A protocol description language for customizing failure semantics. In *Reliable Distributed Systems, 1994. Proceedings., 13th Symposium on*, pages 148–157. IEEE.

[189] Team, B. (2006). Business Motivation Model (BMM) specification. Technical report, Technical Report dtc/06–08–03, Object Management Group, Needham, Massachusetts.

[190] Team, S. et al. (2006). Semantics of business vocabulary and rules (SBVR). Technical report, Technical Report dtc/06–03–02, Object Management Group, Needham, Massachusetts.

[191] Thietart, R.-A. and Forgues, B. (1995). Chaos theory and organization. *Organization science*, 6(1):19–31.

[192] Thomas, M. and McGarry, F. (1994). Top-down vs. bottom-up process improvement. *IEEE Software*, 11(4):12–13.

[193] Tismer, C. (2000). Continuations and stackless Python. In *Proceedings of the 8th International Python Conference*, volume 1.

[194] Tisue, S. and Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Boston, MA.

[195] Tolk, A., Heath, B. L., Ihrig, M., Padilla, J. J., Page, E. H., Suarez, E. D., Szabo, C., Weirich, P., and Yilmaz, L. (2013). Epistemology of modeling and simulation. In *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, pages 1152–1166. IEEE Press.

[196] Tomlinson, C., Kim, W., Scheevel, M., Singh, V., Will, B., and Agha, G. (1988). Rosette: An object-oriented concurrent systems architecture. In *ACM Sigplan Notices*, volume 24, pages 91–93. ACM.

[197] Trevino, L. K. (1986). Ethical decision making in organizations: A person-situation interactionist model. *Academy of management Review*, 11(3):601–617.

[198] Van Cutsem, T., Mostinckx, S., Boix, E. G., Dedecker, J., and De Meuter, W. (2007). Ambienttalk: object-oriented event-driven programming in mobile ad hoc networks. In *Chilean Society of Computer Science, 2007. SCCC'07. XXVI International Conference of the*, pages 3–12. IEEE.

[199] Van Lamsweerde, A. (1991). The KAOS project: Knowledge acquisition in automated specification of software. In *Proc. of the AAAI Spring Symposium Series, Design of Composite Systems, 1991*, pages 59–62.

[200] Van Lamsweerde, A. and Letier, E. (2004). From object orientation to goal orientation: A paradigm shift for requirements engineering. In *Radical Innovations of Software and Systems Engineering in the Future*, pages 325–340. Springer.

[201] van Langevelde, I., Philipsen, A., and Treur, J. (1992). Formal specification of compositional architectures. In *10th European conference on Artificial intelligence*, pages 272–276.

[202] Varela, C. and Agha, G. (2001). Programming dynamically reconfigurable open systems with SALSA. *ACM SIGPLAN Notices*, 36(12):20–34.

[203] Venable, J., Pries-Heje, J., and Baskerville, R. (2012). A comprehensive framework for evaluation in design science research. In *International Conference on Design Science Research in Information Systems*, pages 423–438. Springer.

[204] Vernadat, F. (2002). UEML: Towards a unified enterprise modelling language. *International Journal of Production Research*, 40(17):4309–4321.

[205] Voelter, M. and Lisson, S. (2014). Supporting Diverse Notations in MPS' Projectional Editor. In *GEMOC@ MoDELS*, pages 7–16.

[206] Von Bertalanffy, L. (1968). General system theory. *New York*, 41973(1968):40.

[207] Wakita, K. (1995). First class continuation facilities in concurrent programming language Harmony/2. In *Theory and Practice of Parallel Programming*, pages 300–319. Springer.

[208] Wegmann, A. (1987). Enterprise Architecture. *IBM Systems Journal*, 26(3):276.

[209] White, S. A. (2008). *BPMN modeling and reference guide: understanding and using BPMN*. Future Strategies Inc.

[210] Williams, T. (1998). The Purdue enterprise reference architecture and methodology (PERA). *Handbook of life cycle engineering: concepts, models, and technologies*, Springer Berlin, 289.

[211] Wisnosky, D. E. and Vogel, J. (2004). *DoDAF Wizdom: A Practical Guide to Planning*. Wisdom Systems, Inc.

[212] Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, page 38. ACM.

[213] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

[214] Wolfram, S. (1984). Cellular automata as models of complexity. *Nature*, 311(5985):419.

[215] Wood, M. T. (1973). Power relationships and group decision making in organizations. *Psychological Bulletin*, 79(5):280.

[216] Wooldridge, M. and Jennings, N. R. (1994). Agent theories, architectures, and languages: a survey. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 1–39. Springer.

[217] Yonezawa, A., Briot, J.-P., and Shibayama, E. (1988). Object-oriented concurrent programming in ABCL/1. In *Readings in Distributed Artificial Intelligence*, pages 434–444. Elsevier.

[218] Yu, E., Strohmaier, M., and Deng, X. (2006a). Exploring intentional modeling and analysis for enterprise architecture. *Enterprise Distributed Object Computing Conference Workshops*. doi=10.1109/EDOCW.2006.36.

[219] Yu, E., Strohmaier, M., and Deng, X. (2006b). Exploring intentional modeling and analysis for enterprise architecture. In *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06. 10th IEEE International*, pages 32–32. IEEE.

[220] Zachman, J. et al. (1987). A framework for information systems architecture. *IBM systems journal*, 26(3):276–292.

[221] Zwicky, W. R. (2008). *AJ: A system for building actors with Java*. PhD thesis, University of Illinois at Urbana-Champaign, url: http://osl.cs.illinois.edu/publications/AJ.html.

Appendix A

Review of remaining EM techniques

This research reviews the existing EM techniques using a combination of SMS and SLR methodologies to evaluate their suitability as an aid for organisational decision-making. Chapter 4 presents adopted review methodology, review template, the list of EM techniques reviewed (*i.e.* Table 4.2), review summary of a set of EM techniques (which are extensively referred in this thesis) and the key review findings. The review outcome of the remaining EM techniques, which are referred in IS domain, are discussed in this chapter. Each review outcome contains a brief description of the EM technique and an instance model of the review template, which is described as a meta-model described in Figure 4.5.

Unified Modeling Language (UML)

Unified Modelling Language (UML) [151] is a general purpose extensible modelling language for representing, visualising and documenting the artifacts of software systems. In general, UML supports the structural and behavioural specification of the systems. The structural aspects are captured using seven diagram types: *Class Diagram*, *Component Diagram*, *Composite Structure Diagram*, *Deployment Diagram*, *Object Diagram*, and *Package Diagram*. The behavioural diagrams are - *Activity Diagram*, *Communication Diagram*, *Interaction overview diagram*, *Sequence Diagram*, *State diagram*, *Timing diagram* and *Use case diagram*. In addition to these predefined diagrams, UML supports the concepts of *profile* and *stereotype* for domain specific extensions. The new semantics can be defined using *stereotype*. There are initiatives, such as [153] and [84], where the UML is customised for defining the structural and behavioural concepts of an enterprise using UML profiles and a set of stereotypes.

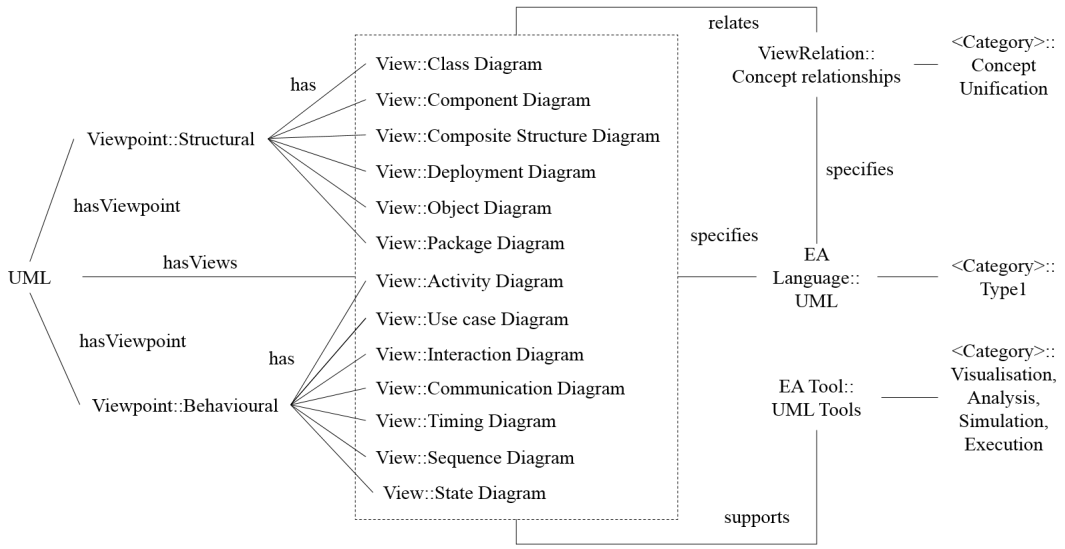


Figure A.1 Instance model of UML

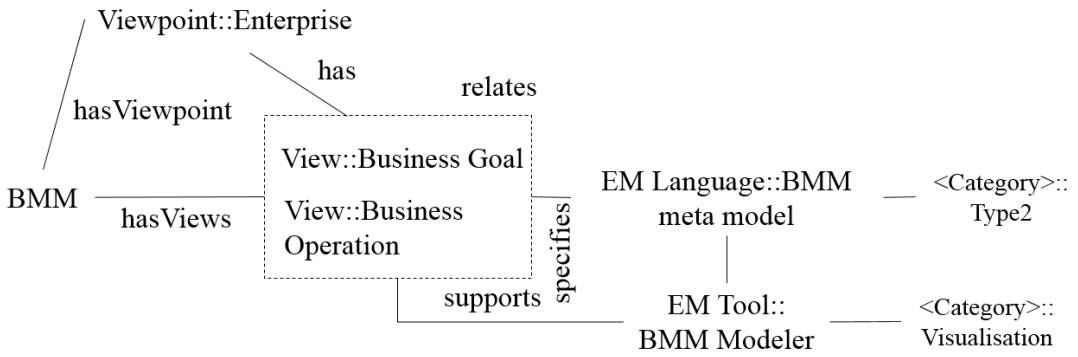


Figure A.2 Instance model of BMM

The SLR on UML produces an instance model of *EM Synthesis* meta-model as depicted in Figure A.1. The UML has two `Viewpoints`: *Structural* and *Behavioural*. The *Structural* `viewpoint` supports six structural diagrams and *Behavioural* `Viewpoint` supports seven behavioural diagrams. The UML has a unified meta-model to establish the `Views` and `ViewRelations`, the UML specification has precise semantics, and the UML centric tools offer a range of visualisation, analysis, simulation and execution capabilities as shown in the figure.

Business Motivation Model (BMM)

Business Motivation Model (BMM) [189] is an OMG[1] standard for specifying and communicating the business plans in an structured form. The Business Motivation Model (BMM)

[1] https://www.omg.org/

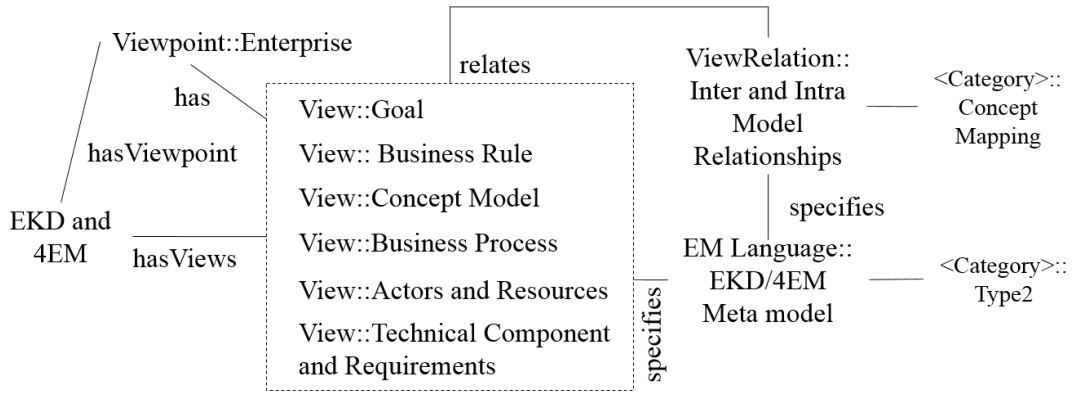


Figure A.3 Instance model of EKD

specification supports the semantics and notations to capture and visualise *Business Goals* and *Business Operations* of enterprises. The model produced from SLR on BMM is depicted in Figure A.2.

Enterprise Knowledge Development (EKD) & For Enterprise Modelling (4EM)

Enterprise Knowledge Development (EKD) [168] is an integrated method and technique to design, analyse, and plan enterprise business. An EKD specification describes *who* does *what*, *how* and *why*. The For Enterprise Modeling (4EM) [173] is the successor of the EKD technique. It supports six different models: *Goal model*, *Process model*, *Actors and resource model*, *Concepts model*, *Business rules model*, and *Technical component model*. The *Goal model* describes *what* people want to achieve in their business, *Process model* describes the flow of activities, *Actors and resource model* describes *who* is involved with the activities, *Concepts model* defines the *things* and *phenomena* of a business, *Business rules model* describes rules that triggers the activities, and *Technical components model* defines the information systems and their use in the enterprise. In a 4EM model, the modelling constructs within a specific models are related using *intra-model relationships* and the constructs across the models are related using *inter-model relationships*. The SLR on EKD and 4EM is depicted in Figure A.3. As shown in the figure, the EKD and 4EM has six `Views`, the `ViewRelations` (*i.e.*, *intra-model relationships* and *inter-model relationships*) are essentially the `Concept Mapping` kinds of relationships, and the EKD/4EM meta-model is a `Type2` specification.

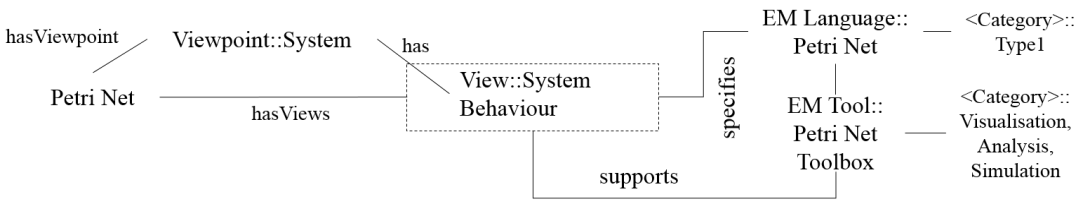


Figure A.4 Instance model of Petri Net

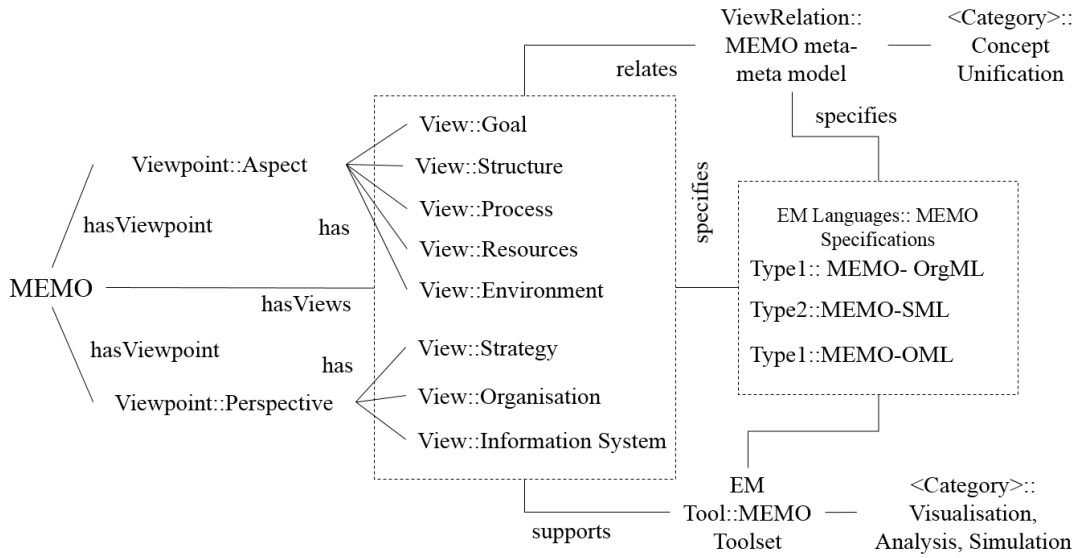


Figure A.5 Instance model of MEMO

Petri Net

Petri Net [156] is a mathematics based modeling language that helps to describe distributed systems using the concepts of *Places*, *Transitions* and *Tokens*. The *Tokens* are the resources that flows, *Places* are the reservoirs of the resources, and the *transitions* consume and produce the resources based on the *firing* rules. The complex workflow and behaviour of a system can be specified using a Petri Net model. The instance model of Petri Net SLR is depicted in Figure A.4. It supports `Type1 EM Language`. The `EM Tools`, such as Petri Net Toolbox[2], help to visualise, analyse and simulate the behavioural aspect of the systems.

Multi-Perspective Enterprise MOdelling (MEMO)

Multi-Perspective Enterprise Modelling (MEMO) [80] is a generic conceptual framework that helps to capture the common abstractions of business firms. It recognises three distinct perspectives - *strategy*, *organization* and *information system* wherein each perspective is structured

[2]https://www.mathworks.com/

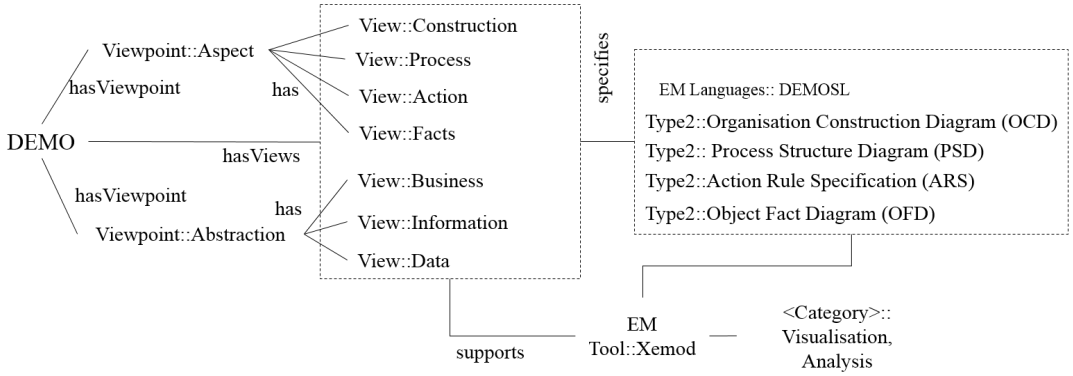


Figure A.6 Instance model of DEMO

along four aspects: *structure*, *process*, *resources*, and *goals*. In addition to these four aspects, the MEMO specification helps to capture the *environmental* aspect of an enterprise. The model produced from the MEMO SLR is depicted in Figure A.5. As shown the figure, MEMO supports two `Viewpoints` - *Aspects* and *Perspective*. The *Aspect* `Viewpoint` has five `Views` and *Perspective* `Viewpoint` has three `Views`. The concepts of these `Views` are specified using a unified meta meta-model, which is known as the MEMO meta meta-model; the `Views` and `ViewRelations` are specified using three `EM Languages`: *MEMO-SML*, *MEMO-OrgML*, *MEMO-OML*. The language MEMO-SML is for strategy modeling; the MEMO-OrgML helps to model the organisation that includes the business processes and resources; and the MEMO-OML is an object oriented modeling language to specify information systems. The associated tools are capable of supporting the visualisation, analysis and simulation as shown in Figure A.5.

Design and Engineering Methodology for Organisation (DEMO)

Design and Engineering Methodology for Organizations (DEMO) [73] is an enterprise modelling technique for transaction and business process modelling. It assumes that an organisation consists of three integrated layers: *B-organization*, *I-organization*, and *D-organization*. The *B-organization* represents the business layer, *I-organization* represents the information layer, and *D-organization* represents the data layer respectively. These three layers form three perspectives or levels of abstraction: business system or *B system*, information system or *I system* and the data system *D system*. These perspectives can be specified using four aspect models: *Construction model* (CM), *Process model* (PM), *Action model* (AM), and *Fact model* (FM). The *Construction model* is the ontological model that describes organisation, organisational units, their internal structure, and the environment where the organisation operates; the *Process model* represents

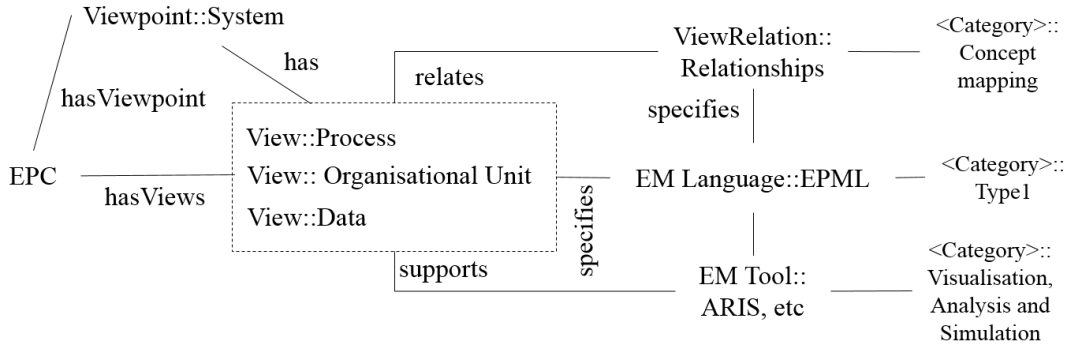


Figure A.7 Instance model of EPC

the state space and the transition space of an organisation; the *Action model* consists of a set of action rules; and the *Fact model* specifies the production rules for state transitions.

The instance model produced from SLR on DEMO is presented in Figure A.6. As shown in the figure, it supports two `Viewpoints`: *Aspects* and *Abstraction*. The *Aspects* `Viewpoint` has four `Views`: *Construction*, *Process*, *Action* and *Facts*. The *Abstraction* `Viewpoint` has three `Views`: *Business*, *Information*, *Data*. There are two ways of representing these `Views`: graphically (*i.e.*, diagrams and tables) and textually. The graphical models, such as *Organisation Construction Diagram* (OCD), *Process Structure Diagram* (PSD), *Action Rule Specification* (ARS), and *Object Fact Diagram* (OFD), are `Type2` category `EM Language`, whereas the textual specification, DEMOSL, is a machine interpretable language thus it is a `Type1 EM Language`. The `EM Tools`, such as Mphee[3] and Xemod[4], are primarily the model visualisation tools.

Event-driven Process Chain (EPC)

Event-driven Process Chain (EPC) [136] is a type of flowchart specification to specify and visualise the business process and workflows. It is a simple and easy-to-understand specification that supports the primitives process concepts that include – *Event*, *Function*, *Information*, *Organisational Unit*, *Process Owner*, *Control Flow* (*e.g.*, alternate, parallel, sequence), *Logical Connectors* (*e.g.*, AND, OR, XOR), and the *Information Flow*. The Event-driven Process Chain (EPC) specification is primarily developed as part of ARIS but later it is extended to use as an independent process modelling specification.

The model produced in SLR is shown in Figure A.7. It supports three `Views`: *Process*, *Organisational Unit* and *Data*. The `EM Language` known as EPML is a `Type1` specification

[3] http://www.mphee.nl/
[4] http://www.ee-institute.org/en/demo/tools

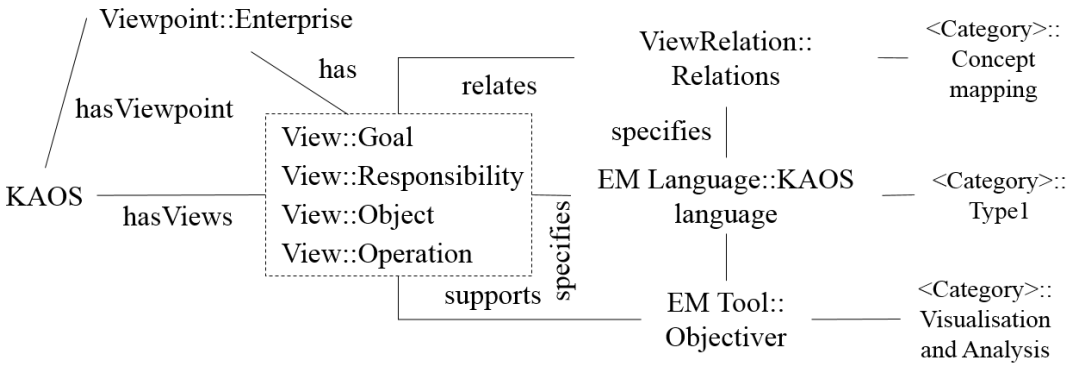


Figure A.8 Instance model of KAOS

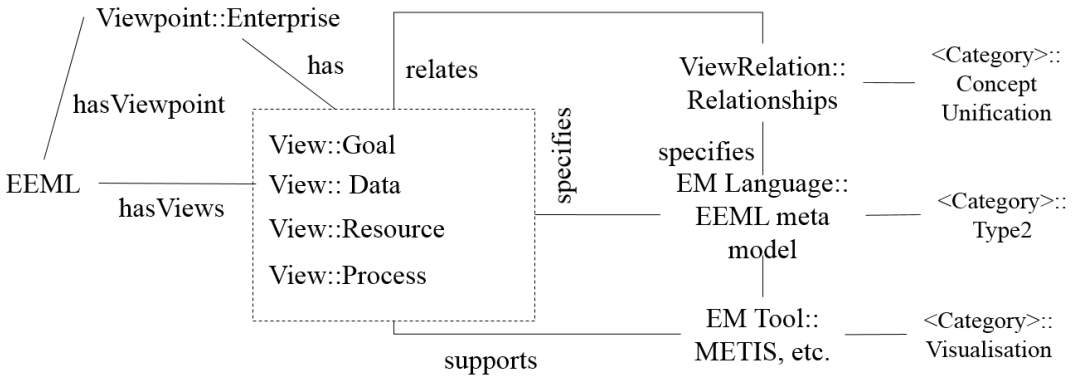


Figure A.9 Instance model of EEML

and `EM Tool`, *e.g.*, ARIS toolset, is capable of supporting visualisation, analysis and simulation of business processes.

Knowledge Acquisition in Automate Specification (KAOS)

Knowledge Acquisition in automated specification or Keep All Objectives Satisfied (KAOS) [199] is a formal goal modelling language developed for requirements engineering. The constructs of KAOS language are capable of describing the *objects*, *operations*, *responsibility* and *goals*. The *objects* represent the things of interest, such as entities, relationships, and events. The *operations* are the input-output relations over objects, which are specified using the pre-, post-, and trigger conditions. The *responsibilities* are the active elements such as humans, devices, software, *etc*. The *goals* are prescriptive statement of the intents of a system and/or responsibilities.

The instance model produced from SLR on KAOS is presented in Figure A.8. The model shows that KAOS has four `Views` - *Goal*, *Responsibility*, *Object* and *Operation*. The KAOS

language is a Type1 EM Language and the associated tool, Objectiver[5], is a visualisation and analysis tool for goal modelling.

Extended Enterprise Modeling Language (EEML)

Extended Enterprise Modelling Language (EEML) [115] is a modelling language that combines structural model, business process model, goal model and resource model. The structural model describes the structure of an enterprise using UML class diagrams; the process modelling describes the process logic through nested structures of tasks and decision points; the resource model describes the roles, *i.e.*, persons, organisations, material objects, software tools and manual tools, in a process model form; and the goal model bridges with other models by associating the intentions of the other model elements. The structural specification is capable of specifying the complex structure, however, the process specification chiefly supports the deterministic process flow.

The outcome of SLR on EEML is depicted in Figure A.9. As shown in the figure, EEML supports four Views and those Views are related through a unified meta-model. The EEML meta-model is a Type1 EM language, and associated tools, such as METIS[6], offer visualisation capability.

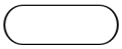
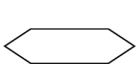
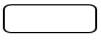
[5] www.objectiver.com/
[6] www.opengroup.org/architecture/0201anah/briefing/computas.pdf

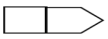
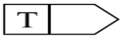
Appendix B

OrgML Notations

OrgML models across this thesis are represented using a set of notations. Those OrgML notations along with the brief descriptions of the OrgML concepts are listed below:

Table B.1 OrgML Notations

	Concept	Description	Notation
Unit Definitions	OrgUnit	Modular, autonomous, reactive unit that represents organisation, organisational units and environment	
	DataUnit	A data structure that represents a collection of `Variables`	
	Calendar	An entity that contains and specifies global `TimeEvents`	
Decision Making Concepts	Goal	Intention or objective of organisational decision-making or `OrgUnit`	
	Measure	A `Variables` that indicate the key performance indicators (of an `OrgUnit`)	
	Lever	A course of action or change that can be applied on an `OrgUnit`	
Behavioural Elements	Action	A behavioural unit with a coherent set of `Statements` that activates when an `Event` specification is satisfied	
	Function	A behaviour unit that contains a coherent set of instruction `Statements`	

	Concept	Description	Notation
Structural Elements	Variable	A *typed* entity that represents characteristic or state variable of an `OrgUnit`	
	Parameter	A specialised `Variable` that helps to characterise an `OrgUnit`	
	Exposed Variable	A `Variable`, which is exposed from an `OrgUnit`	
	Trace	A sequence of `Data` that captures `State`, `Events` produced internally, `Events` communicated to other `OrgUnits`, and `Events` received by an `OrgUnit` along with the time information from a point in time in the past till now.	
Event Specification	Incoming Event	`Event` received by an `OrgUnit`	
	Outgoing Event	`Event` triggered by an `OrgUnit`	
	Internal Event	`Event` internal to an `OrgUnit`	
	Time Event	`Event` that indicate relative time	
Relationships	Containment	Containment relationship of `OrgUnit`	
	Inheritance	`OrgUnit` Inheritance relationship	
	Association	OrgML Association	
	Event communication	Message passing between two `OrgUnits`	
Lever Spec.	Variation Point	Location where a `Lever` can be applied	
	Variation	Alternative for describing `Lever`	

Appendix C

OrgML to ESL translation rules

The transformation rules to transform the OrgML concepts to the ESL concepts are described using Xtend model transformation template language [39]. This chapter introduces Xtend model transformation template language and presents the key OrgML to ESL transformation rules.

C.1 Overview of Xtend model transformation language

Xtext is a text-based functional transformation language from Eclipse[1]. It can transform a model, which is defined using a domain specific meta-model to a model or a textual specification that conforms to a meta-model or grammar. The capabilities of the Xtend transformation specification are described using a case that considers a subset of UML Class diagram to Java class transformation as shown in Figure C.1.

A meta-model that captures *Class* with a set of typed *Properties* is considered as the source meta-model. A transformation template that translates the concept of *Class* and *Property* to Java class is shown in Figure C.1 (b). The transformation template iterates over all model elements in line 2, filters *Class* instances in line 3, generates Java class declaration in line 4, iterates over all *Properties* to generate attribute declarations (in line 5–7) and generate Java attributes declaration in line 12–14. An instance of the source meta-model depicted in Figure C.1 (a) is shown in Figure C.1 (c) and the translated Java code is shown in Figure C.1 (d).

The next section presents the OrgML to ESL transformation rules using Xtend model transformation template language.

[1] http://www.eclipse.org/xtend/documentation/index.html

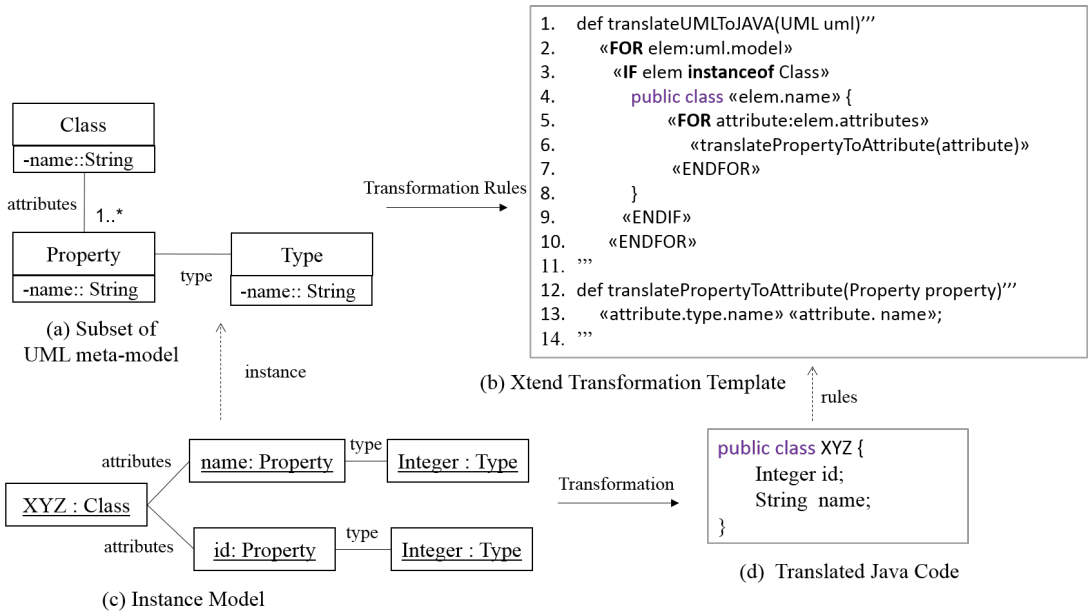


Figure C.1 Overview of Xtend transformation

C.2 OrgML to ESL transformation rules

A schematic representation of OrgML to ESL transformation template is shown using Xtend language in Figure C.2. Template `translateOrgMLToESL` iterates over all `OrgUnits` of a model (line 6) and generates ESL actor specification by navigating and transforming OrgML elements, such as `Parameters`, `Variables`, `Measures`, `Events`, `Functions`, `Actions` and `Levers`, to appropriate ESL constructs as described in Table 5.2.

The transformation rules of the constituent elements of a `OrgUnit` are highlighted in line 8 – 41 of Figure C.2. OrgUnit `Parameters` are translated into ESL actor variables as shown in line 8, 20 and 52–54; all exposed and encapsulated `Variables` are translated to ESL actor variables as shown in line 11, 21, 22 and 63–66; and `Traces` are translated to ESL actor variables as shown in line 65 of Figure C.2. The model navigation rules that are used to define transformation rules are shown in Figure C.3.

The `IncomingEvents`, `InternalEvents` and `TimeEvents` are translated to ESL event specification in line 32–35. OrgML `Functions`, `Actions` and `Levers` are also translated to ESL specification in line 14, 37 and 40–42 respectively.

The transformation rules for OrgML `Action`, `Event`, `Calendar` and inherited `OrgUnit` are described below.

```
1  /*  Legends: Template Specification, ESL Keywords,
2   *  Model Elements, Model Properties, Model Associations of OrgML meta-model
3   */
4
5  def translateOrgMLToESL(OrgM ⊨ orgml)‘‘‘
6   «FOR unit:orgml.model»
7    «IF unit instanceof OrgUnit» //Transform OrgUnit as Actor
8      act «unit.name» («generateParameterList(getAllParameters(unit))») {
9
10       //Transform Exposed Variables as exported variable of an actor
11       export «getAllExposedVariables(unit).map[name].join(',')»
12
13       //Transform Function Specifications
14       «FOR behav:getAllFunctions(unit)»  «transformFunctionSpec(behav)» «ENDFOR»
15
16       //Transform Measures and TraceExpressions
17       «FOR m:getAllMeasures(unit)» «createFunctionForMeasure(m)»  «ENDFOR»
18
19      //Transform Variables and Traces
20       «FOR p:getAllParameters(unit)» «p.name»::«p.vtype.name» := 'p_'«p.name» «ENDFOR»
21       «FOR v:getAllExposedVariables(unit)» «transformVariable(v)» «ENDFOR»
22       «FOR v:getAllEncapsulatedVariables(unit)» «transformVariable(v)» «ENDFOR»
23       «FOR v:getAllMeasures(unit)» «transformVariable(v)» «ENDFOR»
24
25       // Transform EInfo
26       eventTrace::[Event] := []
27
28       //Register for subscribed TimeEvent
29       → { «FOR e:getAllSubscribedEvents(unit)» calendar ← RegieterFor«LectureSlot.name»(self)» «ENDFOR»  }
30
31       //Transform Event Specifications
32       «FOR e:getAllIncomingEvents(unit)» «transformEventSpec(e)» «ENDFOR»
33       «FOR e:getAllInternalEvents(unit)» «transformEventSpec(e)» «ENDFOR»
34       «FOR e:getAllSubscribedEvents(unit)» «transformEventSpec(e)» «ENDFOR»
35
36       //Transform Action Specification as Internal Actor and Events
37       «FOR behav:getAllActions(unit)» «transformActionSpec(behav)» «ENDFOR»
38
39       //Transform Lever Specification as IncomingEvent
40       «FOR lever:getAllLevers(unit)»
41           «lever.name»  → { «transformLeverSpec(lever)» }
42       «ENDFOR»
43
44       //Support Human-in-the-loop Goal Evaluation by displaying the relevant measures
45       «FOR goal:unit.goal»  «FOR leaf:getLeafLevelGoal(goal)» Display «leaf.gm.name»  «ENDFOR» «ENDFOR»
46
47     }
48    «ENDIF»
49
50    «IF unit instanceof DataUnit» // Transform DataUnit as Actor
51       act «unit.name» («generateParameterList(variable:unit.contains)») {
52         «FOR v:unit.contains» «p.name»::«v.vtype.name» := 'p_'«v.name» «ENDFOR» // Variables
53       }
54   «ENDIF»
55  «ENDFOR»
56  '''
57
58 def  generateParameterList(Parameter[] parameters)‘‘‘
59    «parameters.map['p_'+ name :: vtype.name].join(',')»
60  '''
61 def  transformFunctionSpec(Function function)‘‘‘
62   «function.name»(«generateParameterList(function.params)»)::«function.returns»
63           =  «translateBSpec(function.spec)»  //Function Specification
64  '''
65 def createFunctionForMeasure(Measure measure)‘‘‘
66   evaluate_«measure.name»()::«measure.vtype.name»
67           = compute_measure_value(measure.value.uses)
68  '''
69 def  transformVariable(Variable variable)‘‘‘
70    «variable.name»::«variable.vtype.name»
71    trace_«variable.name»::[«variable.vtype.name»] := []
72  '''
73 def  generateArgumentList(Data args)‘‘‘
74   «args=args.contains.map['p_'+ name  vtype.name].join(',')»
75  '''
```

Figure C.2 Overview of OrgML to ESL transformation rules

C.2.1 Transformation rule for OrgML Action, Event and BSpec

Transformation of event specification, `transformEventSpec`, is highlighted in line 9–26 of Figure C.4. The `IncomingEvents`, `InternalEvents` and subscribed `TimeEvents` have a set of `Statements` that can be translated to ESL statements by translating the syntactic differences of OrgML `BSpec` and ESL statement. In addition, each event definition may display measure values (shown in line 14) and capture trace information as shown in line 17 and 18.

```
1  /*
2   * Legends:Model Elements, Model Properties, Model Associations of OrgML
3   */
4   def Parameter[] getAllParameters(OrgUnit unit)  { return unit.params }
5   def Measure[] getAllMeasures(OrgUnit unit) { return unit.measures}
6   def Variable[] getAllExposedVariables(OrgUnit unit) { return unit.exposes}
7   def Variable[] getAllEncapsulatedVariables(OrgUnit unit) { return unit.encapsulates}
8   def IncomingEvent[] getAllIncomingEvents(OrgUnit unit) {unit.receives}
9   def InternalEvent[] getAllInternalEvents(OrgUnit unit) {unit.internal}
10  def TimeEvent[] getAllTimeEvents(OrgUnit unit) { return unit.subscribes}
11  def Function[] getAllFunctions(OrgUnit unit) {
12   var ArrayList<Function>  functions = new ArrayList<Function>()
13   for (element : org.behaviour(org))
14     if (element instanceof Function)
15         functions.add(element)
16    return functions
17  }
18  def Action[] getAllActions(OrgUnit unit) {
19   var ArrayList<Action> actions = new ArrayList<Action>()
20   for (element : org.behaviour(org))
21     if (element instanceof Action)
22         actions.add(element)
23    return actions
24  }
25 def Lever[] getAllLevers(OrgUnit unit) {unit.levers}
```

Figure C.3 OrgML model navigation rules

The translation rules for OrgML `Action` to ESL specification (as illustrated in Figure 5.23) is shown in line 32–66 of Figure C.4). For each `Action`, the transformation specification generates a new inner actor (line 32) with a set of actor elements that include – (i) '*expectedEventTrace*' variable along with the event specification (as shown in Figure 5.23) (line 33), (ii) a variable to capture '*actualEventTrace*' (line 34), (iii) functions to evaluate the event trace (line 36), evaluate state variables (line 37) and perform set of actions (line 43), and (iv) a set of event definitions (line 45–55). The generated event specification stores event parameters so that they can be used by the action statements (line 49), updates '*actualEventTrace*' (line 51), evaluates event trace condition and state variable conditions (line 53), and performs action statements if the event conditions and trace conditions are `true` as shown in line 53.

C.2.2 Transformation rule for OrgML Calendar

OrgML `Calendar` is translated to ESL actor using the transformation rules shown in Figure C.5. The transformation rules generate a set of variables to capture subscribers (line 7–8), a set of event specifications for specified `TimeEvents` (line 9–12), and a specification of the primitive `TimeEvent`, which is termed as `Time`, as shown in line 13–15. The `Time` event computes all derived `TimeEvents` as illustrated in Figure 5.24.

C.2.3 Transformation rule for inherited OrgUnit

The OrgUnit inheritance is resolved by translating inherited `OrgUnits` to ESL actors such that each inherited `OrgUnit` includes its own and inherited `Variables`, `Parameters`, `Events`, `Functions` and `Actions` as discussed in section 5.5. The translation rules shown in Figure C.2

```
/*  Legends: Template Specification, ESL Keywords,
 *  Model Elements, Model Properties, Model Associations of OrgML meta-model
 */
def transformBSpec(BSpec bspec)‘‘‘
«FOR stmt:bspec.stmt»
    //Syntactic transformation of orgml statements
«ENDFOR»
’’’
def transformEventSpec(BehaviouralEvent event)‘‘‘
var arguments = «generateArgumentList(event.carries)»

«event.name»(«arguments») → {
   //Display Measures
   «FOR measure:event.maps» Display evaluate_«measure.name»() «ENDFOR»

   //Capture trace information
   «IF event instanceof TimeEvent»  trace_«variable.name» :=
                trace_«variable.name» + [«variable.holds»] «ENDIF»

   «FOR action:event.used»            //Send messages to all action actors
     for n::Int in 0..(length(variable_«action.name») -1) do
         variable_«action.name»[n] ← event(«arguments»)
   «ENDFOR»

   «translateBSpec(event.attachedTo.spec)»   //Event specification
 }
’’’
def transformActionSpec(Action action)‘‘‘
variable_«action.name»::[«action.name»] :=[] //Variable to refer inner actors

 //Add new inner actor to monitor a complex event
 act act«action.name» ()::Act«action.name» {
  expectedEventTrace::Str = «generateExpectedTrace(action.spec)»
  actualEventTrace::[Str] = []

  eventTraceEvaluator()::Bool = evaluateTrace(expectedTrace,eventTrace)  // Standard trace evaluation logic
  actionConditionEvaluator()::Bool = «generateWhenConditionEvaluation(action.spec)»

  eventArgs::[Str][Data]  //Variable to store event Data.


  //Function to perform action
  perform()::void = «translateBSpec(action.spec)»  //Action statements

  «FOR event:action.uses»
     variable_«event.name»::Int := 0;
     «event.name»(«generateArgumentList(event.carries)») → {
       // Add Event Arguments
       eventArgs := eventArgs + [«event.name», «event.carries»]
       // Add Event to the Event Trace
       actualEventTrace := actualEventTrace + [«event.name»]

       if (eventTraceEvaluator() && actionConditionEvaluator()) then perform() else nothing
     }
  «ENDFOR»
}
’’’

def generateWhenConditionEvaluation(BSpec bspec)‘‘‘
«FOR exp:bspec.state»
    //Syntactic transformation of orgml expression
«ENDFOR»
’’’
def generateExpectedTrace(BSpec bspec) {
 //Generate expected trace using recursive depth-first search of BSpec.events specification
}
```

Figure C.4 Transformation of Action, Event and BSpec

```
/*  Legends: Template Specification, ESL Keywords,
 *  Model Elements, Model Properties, Model Associations of OrgML meta-model
 */

//OrgML Calendar to ESL Actor translation rules
 act calendar()::Calendar {
   «FOR time:Calender.contains» «time.name»::Int := 0;
          «time.name»Subscriber::[T] = [] «ENDFOR»
     «FOR time:Calender.contains» «time.name» → {
        «time.name» := «time.name» + 1
        for n::Int in 0..(length(«time.name»Subscriber)-1) do nth(«time.name»Subscriber,n) ← «time.name»
     } «ENDFOR»
     Time(primitive::Int)→ {
        «FOR time:Calender.contains» «transformTimeSpecification(time)» «ENDFOR»
     }
 }
```

Figure C.5 Transformation of Calendar

and the model navigation rules shown in Figure C.6 are used to generate ESL actor specification

```
/*
 * Legends: Model Elements, Model Properties, Model Associations of OrgML meta-model
 */

 def OrgUnit[] traverseHierarchy(OrgUnit[] list, OrgUnit unit)
  for (reln:unit.source)
    if (reln instanceof Inheritance)
        traverseHierarchy(list.add(reln.target), reln.target)
  return list

 def Parameter[] getAllParameters(OrgUnit unit)  {
  var HashMap<String,Parameter> map = new HashMap<String,Parameter>()
  for (org: traverseHierarchy([unit],unit))
     for (param : org.parameters(org))
      if (!map.containsKey(param.name))
          map.put(param.name,param)
      else {
          Parameter p= map.get(param.name)
          if (p.type.name != param.type.name)
             error("Error: Parameter overriding is not supported")
      }
   return map.values()
 }

 // Below methods are same as Parameter, i.e., overriding is not supported
 def String[] getAllExportedVariables(OurUnit unit) { ... }
 def Measure[] getAllMeasures(OrgUnit unit) { ... }
 def Variable[] getAllExposedVariables(OrgUnit unit) { ... }
 def Variable[] getAllEncapsulatedVariables(OrgUnit unit) { ... }

 // Collection of all subscribed TimeEvent
 def TimeEvent[] getAllTimeEvents(OrgUnit unit) {
   var ArrayList<TimeEvent>  elements = new ArrayList<TimeEvent>()
   for (org: traverseHierarchy([unit],unit))
 for (t :org.subscribes)
             if (!elements.contains(t))
       elements.add(t)
    return elements
 }

 def Function[] getAllFunctions(OrgUnit unit) {
  var HashMap<String,Function> map = new HashMap<String,Function>()
  for (org: traverseHierarchy([unit],unit))
     for (function : org.behaviour(org))
      if (function instanceof Function)
        if (!map.containsKey(function.name))
          map.put(function.name,function)
        else {
          Function f= map.get(param.name)
          if (!sameFunctionParameter(f, function))
              map.put(function.name,function) // Overloading
           else { } //Overriding
        }
   return map.values()
 }

 // Below methods are same as Function, i.e., overriding and overriding are supported
 def IncomingEvent[] getAllIncomingEvents(OrgUnit unit) { ... }
 def InternalEvent[] getAllInternalEvents(OrgUnit unit) { ... }

 def Action[] getAllActions(OrgUnit unit) {
  var HashMap<String,Action> map = new HashMap<String,Action>()
  for (org: traverseHierarchy([unit],unit))
     for (action : org.behaviour(org))
      if (action instanceof Action)
        if (!map.containsKey(action.name))
          map.put(action.name,action)
        else {} //Always override the inherited action
   return map.values()
 }
 // Below method is same as Action, i.e., always override
 def Lever[] getAllLevers(OrgUnit unit) { ... }
```

Figure C.6 Navigation Rules for overriding and overloading

for an inherited `OrgUnit`. The model navigation rules shown in Figure C.6 conform to the overriding and overloading rules described in section 5.5.

Appendix D

An experiment with Akka

This research adopts actor/agent technology as an underlying simulation engine for quantitative *what-if* analysis. The proposed approach uses ESL [54] to benefit from ESL specific advancements, *i.e.* uncertainty and notion of 'time'. However, this research argues that any other actor language, such as Akka [5] and Erlang [12], can be used with the proposed OrgML based approach. In this context, one of the most prominent industry-scale actor language, Akka, is evaluated using a subset of University case study (presented as a running example in this thesis).

The key objectives to evaluate Akka are two-fold – (i) justify the use of existing actor language as underlying simulation specification in the proposed approach (which is presented in Chapter 5), and (ii) establish a high-level transformation path from OrgML to Akka as a validation of the claim.

This chapter presents an overview of Akka in section D.1. It introduces an OrgML model for the experimentation, discusses experimentation steps and reports observations along with a comparative analysis with respect to ESL in section D.2. A transformational path from OrgML to Akka is presented in section D.3.

D.1 A brief overview of Akka

As discussed in Chapter 4, Akka [5] is an industry-scale *actor* library, which is developed using Java and Scala platform and runs on one or multiple Java Virtual Machines (JVMs). Akka supports `actor` [2, 96] abstraction for distributed and concurrent computing model and uses non-blocking asynchronous messaging over a lightweight event-driven communication

Student

```
1.  public class Student extends UntypedActor {
2.      public Sting name;
3.      private ActorRef  academicToReach;
4.      private String      isQueryResolved = false;
5.      public Student(String name, ActorRef  acad) {
6.                academicToReach = acad;
7.                academicToReach().tell(new String("Query….."), getSelf())
8.      }
9.      static public Props props(String name, ActorRef  acad) {
10.         return Props.create(Student.class, () → new Student(name, acad));
11.     }
12.     private interpretResponse(String response) {
13.         //evaluate response and update isQueryResolved
14.     }
15.     @Override
16.     public void onReceive(Object message) {
17.       if ((message instanceof String) && (message.startsWith("Response") {
18.           interpretResponse(message)
19.           if (! isQueryResolved)
20.             sender().tell(new String("Query….."), getSelf())
21.       }
22.     }
23. }
```

Academic

```
1.  public class Academic extends UntypedActor {
2.    public String name;
3.    public Academic(String name,) {…}
4.    static public Props props(String name) {
5.          return Props.create(Academic.class, () → new Academic(name));
6.    }
7.    @Override
8.    public void onReceive(Object message) {
9.       if ((message instanceof String) && (sender() instanceof Student))
10.           sender().tell(new String("Response….."), getSelf())
11.     }
12. }
```

Main

```
1.  public static void main(String[] args) {
2.     final ActorSystem system = ActorSystem.create("Conversation");
3.     ActorRef academic = system.actorOf(Academic.props("Academic1"));
4.     ActorRef student =
                system.actorOf(Student.props("Student1", academic));
5.  }
```

Figure D.1 Illustration of Akka concepts and APIs

processes for interactions. The key abstractions and APIs supporting these capabilities are listed below:

- `ActorSystem`: It is an environment that creates, manages and executes `actors`.
- `UntypedActor`: An abstract class that can be `extended` to realise an `actor`. Each `UntypedActor` has a `mailbox` to store incoming messages and maintains a thread for computation. Each class that extends `UntypedActor` needs to override `onReceive(...)` method to consume messages from its internal mailbox, and it can send messages to other `actors` using `tell(...)` method.
- `Props`: It helps to parameterise an `actor` though the '`new`' construct.
- `ActorRef`: A reference to an actor.

These core capabilities are illustrated using a simple example that consider an asynchronous conversation (lets consider through e-mail) between an academic and a student where the student raises a query, academic responds and this conversation continues till student's query is resolved.

The code fragments of `Student` class, `Academic` class and a `Main` that creates an actor system are shown in Figure D.1. The `Main` class creates an `ActorSystem` termed as '*Conversation*' as shown in line 2 of `Main` class. It creates an `Academic` class in line 3 and a `Student` class in line 4. The `Student` class and `Academic` class both extend the class `UntypedActor`, implement `Props` method, and override `OnReceive(Object message)` method as shown in the figure. The `Student` initiates a conversation with `Academic` by sending a query to academic

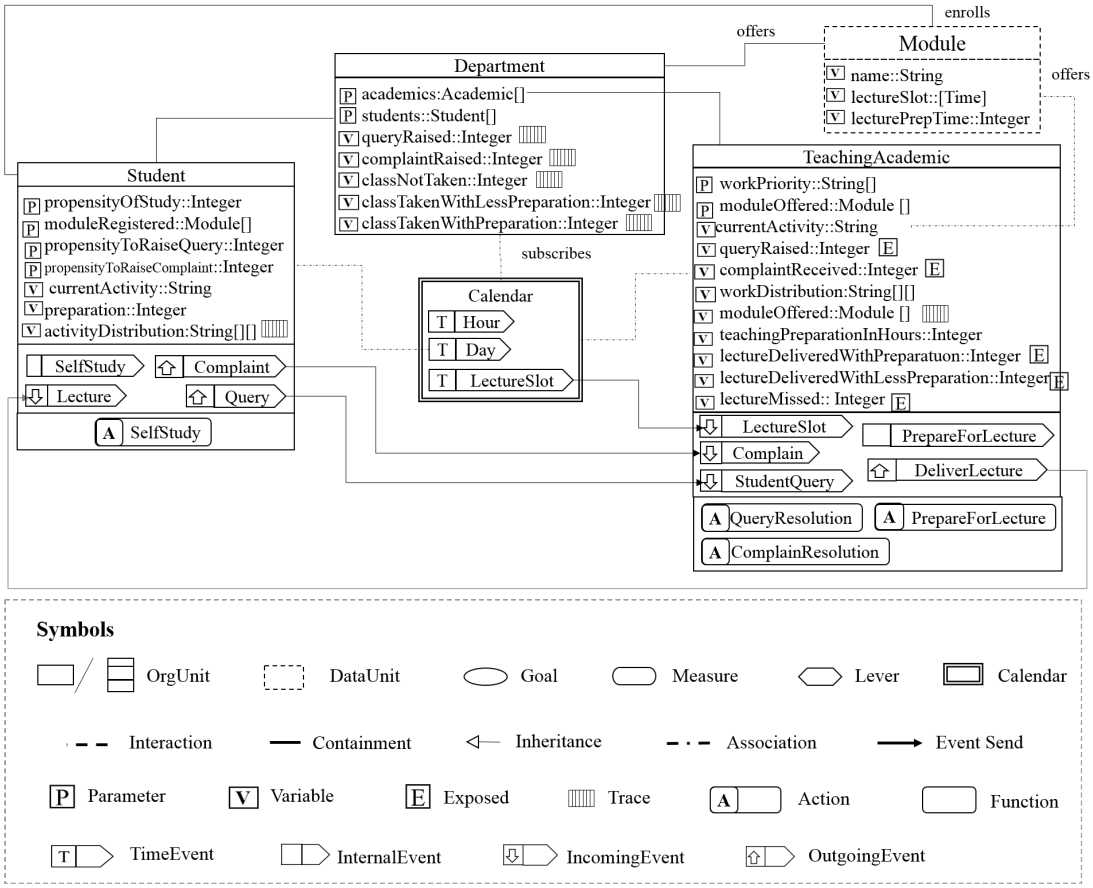


Figure D.2 A subset of University case study

using `tell(...)` as shown in line 7 of `Student` class. Academic identifies student's query by a pattern matching (as shown in line 9 of `Academic` class) and responds back to the sender with a specific response (shown in line 10 of `Academic` class). On the other hand, `Student` finds a response (line 17 of `Student` class), checks if it resolves the query (in line 18 of `Student` class) and raises further query (line no 20 of `Student` class) if query is not resolved. This conversation shows the core capabilities of Akka that will be used in the experiment presented in the next section.

D.2 Experiment

D.2.1 Experimental model

A subset of University case study as shown using an OrgML specification in Figure D.2 is considered for this experiment. The subset includes *Department*, *TeachingAcademic* and *Student* from the case study discussed in section 7.3. *Department* contains *TeachingAcademics* and

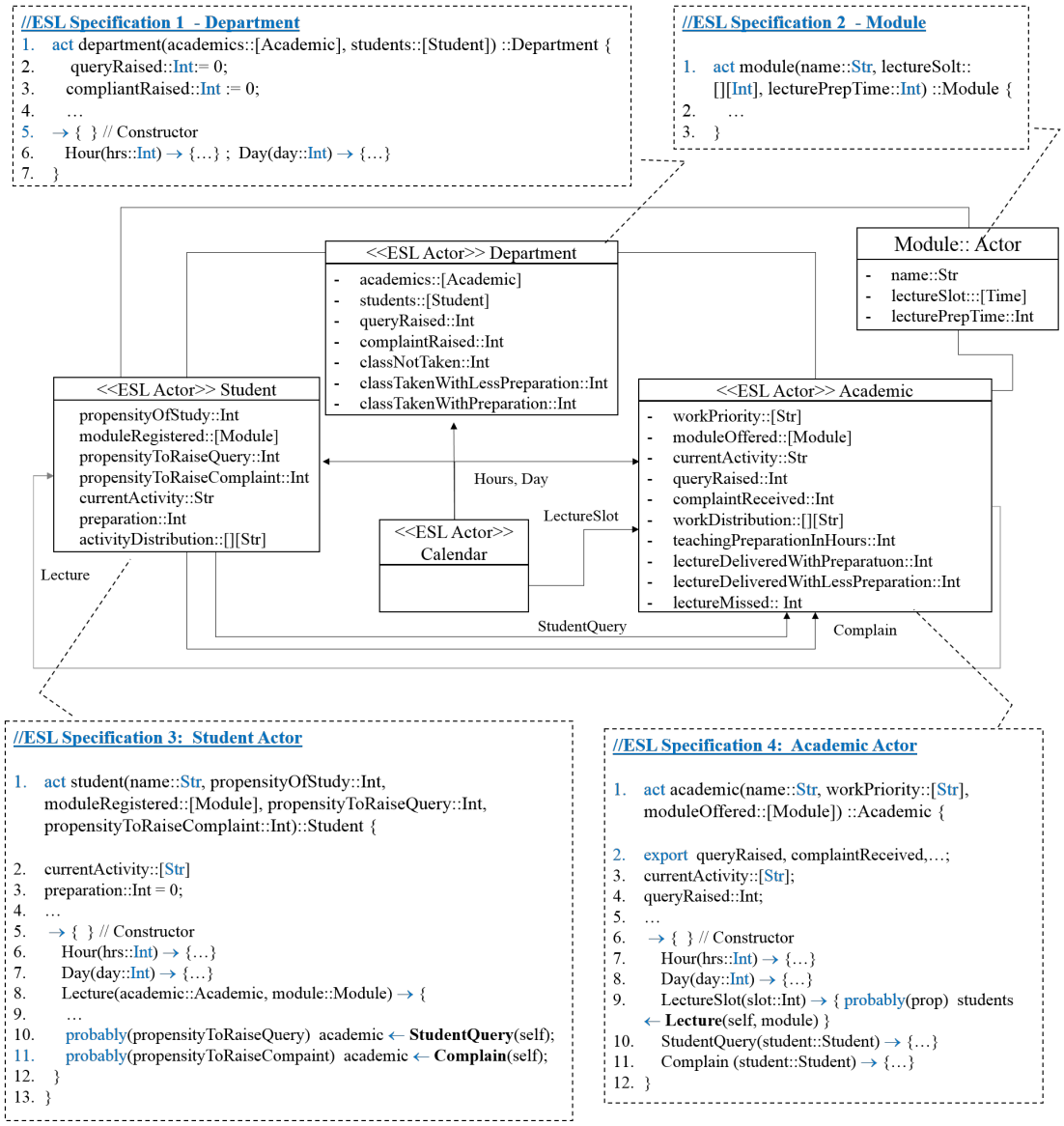


Figure D.3 A schema and sample specification of ESL implementation

Students where *TeachingAcademics* offer a set of *Modules* and *Students* enroll *Modules*. *Department*, *TeachingAcademic* and *Student* subscribe a *Calendar* that recognises '*Hour*', '*Day*' and '*LectureSlot*' `TimeEvents`. The `Parameters`, `Variables`, `Trace`, `IncomingEvents`, `OutgoingEvents`, `InternalEvents` and `Actions` of *Department*, *TeachingAcademic* and *Student* are shown using OrgML notations in Figure D.2. The subset of the `OrgUnit` behaviours that are considered are:

- *Calendar* notifies '*LectureSlot*' `TimeEvent` to *TeachingAcademics*.

- A *TeachingAcademic* may deliver a '*Lecture*' on a specific '*LectureSlot*'. Delivering a '*Lecture*' on scheduled '*LectureSlot*' depends on a probability factor and the priorities for other activities (*i.e.* '*workPriority*'), such as managerial work and unplanned work.
- *Student* may attend a '*Lecture*' (based on a probability factor).
- *Students* may raise a '*StudentQuery*' and/or a '*Complaint*' after attending a '*Lecture*' based on their characteristics, which are specified using parametric variables: '*propensityToRaiseQuery*' and '*propensityToRaiseComplaint*'.

Detailed description of the depicted `OrgUnits` and *Module* `DataUnit` can be found in section 7.3.

D.2.2 Implementation using ESL and Akka

OrgML model shown in Figure D.2 is translated to ESL and Akka for the following purposes – (i) compare ESL and Akka as a simulation specification for OrgML based approach, and (ii) develop a transformation path from OrgML to Akka so that other researchers choose from ESL and Akka for *what-if* analysis needed for organisational decision-making.

ESL Specification

Proposed OrgML to ESL translation rules translate OrgML specification into ESL specification as shown using an extended form of class diagram in Figure D.3. The «ESL Actor» stereotype represents ESL actors, associations represents ESL events or message passing and call-out boxes show the high-level actor specification using ESL.

As shown in the figure, ESL specification contains five interacting ESL *Actors* to represent `OrgUnits`, `DataUnits` and `Calendar`. Precisely, *Department*, *Academic* and *Student* `OrgUnits` are translated into ESL *Actors* where `OrgUnit Parameters` are translated into ESL *Actor* parameters (as shown in '*ESL specification 3*' and '*ESL specification 4*' of Figure D.3), `Variables` are mapped to *Actor* variables with equivalent ESL types, exposed variables are 'exported' from ESL *Actors* (as shown in line 2 of '*ESL specification 4*' in Figure D.3), and the behaviours of all `IncomingEvents`, `InternalEvents` and subscribed `TimeEvents` are translated into ESL event specification. The uncertainty and probabilistic behaviors are speciefied using ESL '`probably`' construct (as shown in lines 10 and 11 of '*ESL specification 3*' and line 9 of '*ESL specification 4*'). The `OutgoingEvents` are raised appropriately from the behavioural

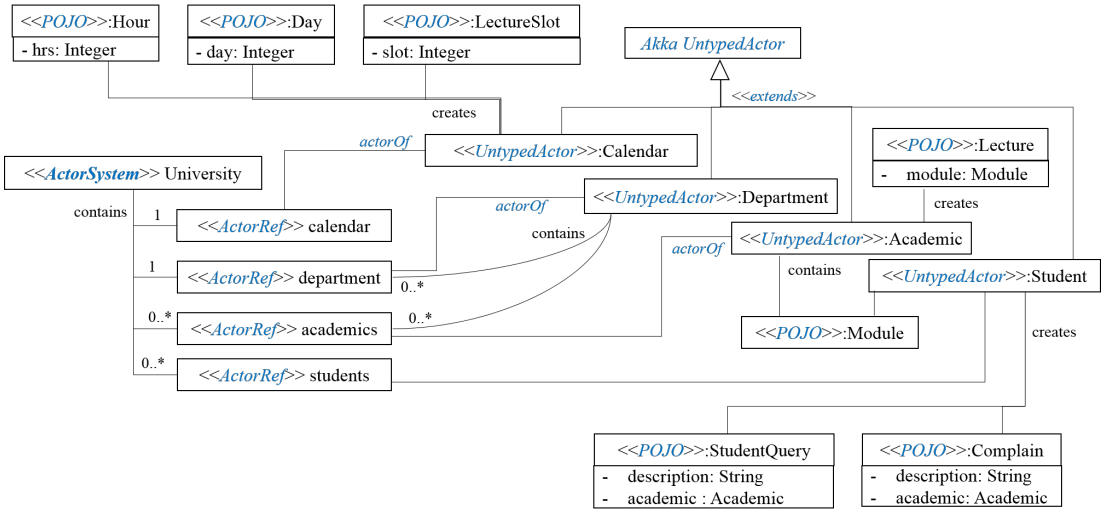


Figure D.4 A schema of Akka implementation

specification (as shown in lines 10 and 11 of '*ESL specification 3*' and line 9 of '*ESL specification 4*'). *Department* actor represents a composite `OrgUnit` that observes and controls *Academic* and *Student* `OrgUnits`, therefore it has minimum behaviour as shown in '*ESL specification 1*' of Figure D.3. It is expected that *Department* behaviour emerges from the interactions of *Academics* and *Students*.

The *Module* `DataUnit` is translated to ESL *Actor*, which has no behaviour as shown in '*ESL specification 2*'. The `Calendar` is also translated to ESL *Actor* that senses primitive event (*i.e.* a'*tick*' raised by ESL simulation engine), computes all composite `TimeEvents` and sends `TimeEvents` to respective `OrgUnits` based on `subscribe` relationships.

Akka Specification

An equivalent Akka specification of OrgML model shown in Figure D.2 is depicted using a class diagram in Figure D.4. Akka implementation classes and interfaces[1], such as `ActorSystem`, `ActorRef` and `UntypedActor`, as well as Java POJO classes are shown using stereotypes in Figure D.4. As shown in the figure, the complete specification is realised as a `ActorSystem` that contains the placeholders (*i.e.* `ActorRef`) for all actors. The `OrgUnits` and `Calendar` are implemented by extending Akka `UnitypedActor` (they can also be represented by extending Akka `AbstractActor`). The *Events* are represented as POJO classes that contains all event parameters as attributes, and the event interactions are realised by overriding `onReceive(...)` and invoking `tell(...)` methods.

[1] https://doc.akka.io/docs/akka/2.5/guide

```
1.  public class Department extends UntypedActor {

2.  ArrayList<ActorRef> academics;
3.  ArrayList<ActorRef> students;
4.  Integer queryRaised;
5.  Integer complaintRaised;
6.  …
7.   public Department(String name, Integer numberOfAcademics, Integer numberOfStudents) {
8.     for (int i=0; i< numberOfAcademics ;i++)  academics.add(system.actorOf(Academic.props(…));
9.     for (int i=0; i< numberOfStudents ;i++)  students.add(system.actorOf(Student.props(…));
10.  }
11.   static public Props props(String name, Integer numberOfAcademics, Integer numberOFStudent) {
12.       return Props.create(Department.class, () → new Department(name, numberOfAcademics, numberOfStudents));
13.   }
14. }
```

Figure D.5 Akka specification to represent Department OrgUnit

```
1.  public class Academic extends UntypedActor {
2.      private String      workPriority[];
3.      private Module   moduleOffered[];
4.      private String      currentActivity;
5.      public Integer     queryRaised;
6.      public Integer     complaintReceived;
7.   …
8.  public Academic(String name, String workPriority[], Module moduleOffered[]) {…}
9.  static public Props props(String name, String workPriority[], Module moduleOffered[]) {
10.       return Props.create(Academic.class, () → new Academic(name, workPriority, moduleOffered));
11. }
12. @Override
13. public void onReceive(Object message) {
14.    if (message instanceof Hour) {…}     // Perform Hourly activities
15.    if (message instanceof Day) {…}      // Perform Daily activities
16.    if (message instanceof LectureSlot) {
17.          if ((Math.random() * 100) < prob))
18.                <<loop for all student>> student.tell(new Lecture(module), getSelf())
19.     }
20.    if (message instanceof StudentQuery) {…}
21.    if (message instanceof Complain) {…}
22.    else { unhandled(message) }
23.  }
24. }
```

Figure D.6 Akka specification to represent Academic OrgUnit

Conceptually, the OrgML model is translated to an `ActorSystem`, which is termed as *University*. It contains `ActorRef` of extended `UntypedActors` that represent *Department*, *Academic* and *Student* `OrgUnits` and `Calendar`. *Module* `DataUnit` is realised as Java POJO class. All `TimeEvents` of `Calendar` (*i.e.* '*Hour*', '*Day*' and '*LectureSlot*'), `OutgoingEvent` of *Academic* (*i.e.*,'*Lecture*'), and `OutgoingEvents` of *Student* (*i.e.*, '*StudentQuery*' and '*Complaints*') are represented using POJO classes as shown in Figure D.5. The Akka specification of *Department*, *Academic*, *Student* and *Calendar* are respectively shown in Figure D.5, D.6, D.7 and D.8.

```
1.  public class Student extends UntypedActor {
2.   private String     name;
3.   private Integer   propensityOfStudy;
4.   private Module   moduleRegistered [];
5.   private Integer   propensityToRaiseQuery;
6.   private Integer   propensityToRaiseComplaint;
7.   …
8.  public Student(String name, Integer propensityOfStudy,Module[] moduleRegistered, Integer propensityToRaiseQuery, Integer
          propensityToRaiseComplaint) {…}
9.  static public Props props(String name, Integer propensityOfStudy,Module[] moduleRegistered, Integer propensityToRaiseQuery,
                    Integer propensityToRaiseComplaint) {
10.       return Props.create(Student.class, () →  new Student(name, propensityOfStudy, moduleRegistered,
                                    propensityToRaiseQuery, propensityToRaiseComplaint));
11.  }
12.  @Override
13.  public void onReceive(Object message) {
14.    if (message instanceof Lecture) {
15.          Lecture lecture = (Lecture) message;
16.          … // behaviour
17.          if ((Math.random() * 100) < propensityToRaiseQuery))
18.                getSender().tell(new StudentQuery(desc), getSelf());
19.           if ((Math.random() * 100) < propensityToRaiseCompalint))
20.                getSender().tell(new Complain(desc), getSelf());
21.    } else {  unhandled(message) }
22.   }
23.  }
```

Figure D.7 Akka specification to represent Student OrgUnit

```
1.   public class Calendar extends UntypedActor {
2.   ArrayList<ActorRef> hourSubscribers;
3.   ArrayList<ActorRef> daySubscribers;
4.   ArrayList<ActorRef> lectureSlotSubscribers;
5.   public Calendar() {
6.       getContext().system().scheduler().schedule(Duration.Zero(), // initial delay
                          Duration.create(1, TimeUnit.MILLISECONDS), // tick interval
                          getSelf(), "Tick", // Primitive TimeEvent
                          getContext().system().dispatcher(), ActorRef.noSender());
7.   }
8.   static public Props props() {
9.          return Props.create(Calendar.class, () →  new Calendar());
10.  }
11.  public void Subscribes(String timeEvent, ActorRef instances [] ) {
12.       //Add instances to appropriate list
13.  }
14.  @Override
15.   public Receive createReceive() {
16.          return receiveBuilder().matchEquals("Tick", x -> {
17.                … // compute composite time events and send to the respective subscribers
18.       }).build();
19.   }
20.  }
```

Figure D.8 Akka specification for Calendar

As shown in Figure D.5, *Department* class `extends` Akka `UntypedActor` class (line 1). It maintains a lists of `ActorRef` of all academic and student instances (as shown in lines 2 and 3) and contains all `OrgUnit` variables as class attributes (*i.e.* lines 4,5 and 6). The constructor forms the department object by creating academic and student instances as shown in line 7–12.

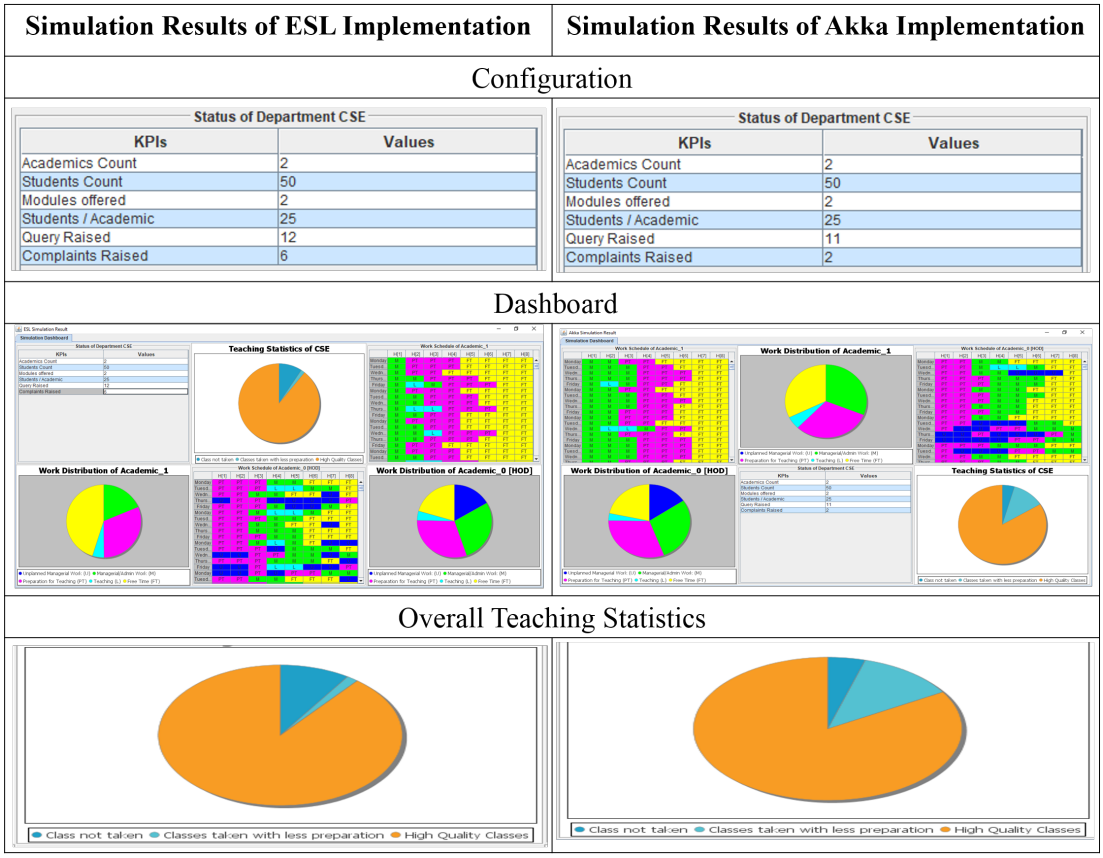


Figure D.9 Simulation results of ESL and Akka

The implementation of *Academic* class, shown in Figure D.6, contains all parameters, variables and traces as shown in line 2–7. It implements `onReceive(Object message)` message to handle subscribed `TimeEvents` (*i.e.*, '*Hour*', '*Day*' and '*LectureSlot*') and all `IncomingEvents`, such as '*StudentQuery*' and '*Complaint*' using pattern matching of incoming `message` as shown in line 12–23.

The attribute access control (*i.e.* `private` or `public`) is set based on the OrgML variable properties, *i.e.* encapsulated or exposed variable, as shown lines 2–6; the uncertainty in delivering a lecture by an academician is implemented using a '*random*' function as shown in line 17; and an example of raising an event using `tell(...)` method is shown in line 18. The *Student* class shown in Figure D.6 implements student `OrgUnit`. Similar to *Academic* class, the *Student* class contains all attributes that represent `OrgUnit` parameters and variables, constructor and implementation of `onReceive(Object message)` method.

Implementation of `onReceive` method pattern matches '*Lecture*' event, processes its behaviour, and raises '*StudentQuery*' and '*Complaint*' based on their propensity as shown in

lines in 17–20. An implementation class for *Calendar* as an extended `UntypedActor` class as shown in Figure D.8. Implemented *Calendar* class contains a set of references to capture subscriptions as shown in lines 2–4, provides an implementation of a primitive `TimeEvent`, which is termed as '*Tick*', as shown in line 6, exposes a method to subscribe `TimeEvents` and provides an implementation to raise all `TimeEvents` by overriding `createRecieve()` method as shown in lines 14–19.

D.2.3 Simulation using ESL and Akka

The translated ESL and Akka specifications of OrgML model (shown in Figure D.2) are simulated for a department configuration that has two academics, fifty students and offers two modules. The simulation results are visusalised using OrgViz Data visualiser by integrating Akka implementation with OrgViz Data visualiser. As shown in Figure D.9, the outputs from ESL and Akka implementation are nearly identical with minor deviations due to the probabilistic nature of the `OrgUnit` or actor behaviours.

D.2.4 Synthesis

The experiment presented in this section considers an OrgML model, translates OrgML model into Akka specification alongside ESL, and shows Akka and ESL based simulation results to evaluate Akka as a simulation specification (and its JVM based execution as simulation engine) in the proposed OrgML based approach (presented in Figure 5.7 of Chapter 5). This experiment demonstrates the applicability of Akka as an alternate simulation specification as it can express most of the requirements of organisational decision-making (*i.e.* requirements listed in Table 3.3) excluding uncertainty, notion of 'time' , goal, measure and lever.

However, the comparison with ESL specification shows that ESL is better suited for organisational decision-making as compared to Akka. The principal reasons are: (i) ESL explicitly specifies uncertainties and supports the notion of primitive 'time', (ii) ESL event specification is more expressive than the Akka specification as an Akka specification expects distinct static Plain Old Java Object (POJO) class for each event as shown in Figure D.4, and (iii) `IncomingEvent` and subscribed `TimeEvents` implementations expect pattern matching code in overridden `OnReceive` method as shown in Figure D.6. Whereas the event specification in ESL is better structured as follows:

`<event name>` (parameter list) $\rightarrow$ `{event spec }` (as shown in Figure D.3)

Table D.1 Mapping from OrgML to Akka

OrgML Concept	ESL Concept	Akka Concept	Akka Sample Code
OrgUnit	Actor	Java Class that *extends* **AbstractActor** or **UntypedActor**	public static class <name> extends UntypedActor {...}
DataUnit	Actor	Static POJO Class	public static class <name> {...}
Parameter	Actor Variables and Input parameter in 'new' operator of ESL Actor	Java attribute and parameter to object 'new'	public <name>(<parameter list>) {…} static public Props props(<parameter list>) { return Props.create(<name>.class, () → new <name>(<parameter list>))}
Encapsulated Variable	Actor Variable	Private attribute with corresponding Java type	private <type> <name>
Exposed Variable	Exported Actor Variable	Public attribute with Java type	public <type> <name>
Trace	Actor Variable	Class Attribute of List Type	public ArrayList<type> <name>
IncomingEvent	Event	Override ***OnReceive(..)*** method	public void onReceive(Object message) { if (message instanceof **<event class>**){…} …}
OutgoingEvent	***send*** Event	***tell()*** with 'new' POJO class	<target instance>.tell(new <event class> (<parameter list>), getSelf())
TimeEvent	Event	Custom API for primitive time and ***OnReceive*** implementation	Same as IncomingEvent
InternalEvent	Event	Java Method	<object name>.<method name>(parameters)
Function	Function	Java Method	public <type> <name>(<parameters>) {…}
Action	Actor, Event and Function	Conditions and Java Method	Java specification
Calendar	Actor	Java Class that extends ***AbstractActor*** or ***UntypedActor***	public static class Calendar extends UntypedActor {...}
Inherited OrgUnit	Actor	Inherited Java class	public static class <name> extends <Base class> {...}
Uncertainty	Probably (probability) ← {statement}	Java expression such as *random* number	if ((Math.random() * 100) < probability)) {statement}
Goal	ESL Specification	Java Specification	-
Measure	ESL Specification	Java Specification	-
Lever	ESL Specification	Java Specification	-

Next section defines a transformation path from OrgML to Akka so that other research can choose from two alternatives – (a) ESL, which is better suited for this research but not yet accepted in industry and (ii) Akka, which is prominent in industry but involves significant effort to develop/generate simulation code. It also requires suitable implementation for 'time' and case specific uncertain behaviour.

D.3 OrgML to Akka transformation

This section presents an one-way model transformation strategy to transform OrgML model into Akka specification. The conceptual mapping from the OrgML concepts to Akka specifications alongside ESL mapping are shown in Table D.1. Conceptually, an `OrgUnit`, its specialisation (*i.e.*, `Organisation` and `Environment`) and `Calendar` can be realised using Java class that `extends` *UntypedActor* or *AbstractActor*. `DataUnit` can be realised using Java class. The interaction between `OrgUnits` can be specified using Java class and `OnReceive(...)` and `tell(...)` methods. Each OrgML *Event* expects a static Java class that captures event parameters as class attributes as shown in Figure D.4. These classes are instantiated and sent to the destination using `tell(...)` method for an interaction as shown in Figure D.7 (*see* '*StudentQuery*' specified in line 18 as an example). The Akka actors process these events using `OnRecieve(...)` method as shown Figure D.6 (*see* '*StudentQuery*' as shown in line 20).

As shown in the table, an OrgUnit `Parameters` can be translated to class attributes and parameters to 'new' class. All exposed and encapsulated `Variables` can be translated in `public` and `private` class attributes. An OrgML `Trace` requires a class attribute with list type. Each `OutgoingEvent` requires a Java class to represent event and event parameters. An event can be sent to the destination using `tell(...)` method. The `IncomingEvents` and subscribed `TimeEvents` need to be mapped to pattern matching cases in `OnReceive(...)` method. All `InternalEvents` can be realised using methods of Java classes. The OrgML `Function` and `Action` specifications transform the OrgML variable assignment, conditional statement, loop, instantiation of new objects syntax (as shown in Figure 5.14) into Java syntax.

D.4 Summary

The experiment presented in this chapter demonstrates that both, ESL and Akka, are suitable for the proposed organisational decision-making approach. However, the use of Akka leads to certain additional accidental complexities [120] as opposed to ESL. A strategy to overcome those accidental complexities and use as a simulation specification in the proposed approach are shown by defining a transformation path from OrgML to Akka. The other Actor/agent language, such as Erlang, can also be evaluated to establish their suitability. The exploration of other Actor/Agent languages is one of the areas for future work of this thesis.

Appendix E

Business Process Outsourcing case study

Business Process Outsourcing (BPO) [33] is a method of subcontracting organisational business process to a third-party organisation for specific purpose such as cost and efficiency. The case study discussed in this chapter considers a decision making scenario from BPO business where a third-party organisation would like to compete with its competitors by offering best in class, value added and economical services to its customers.

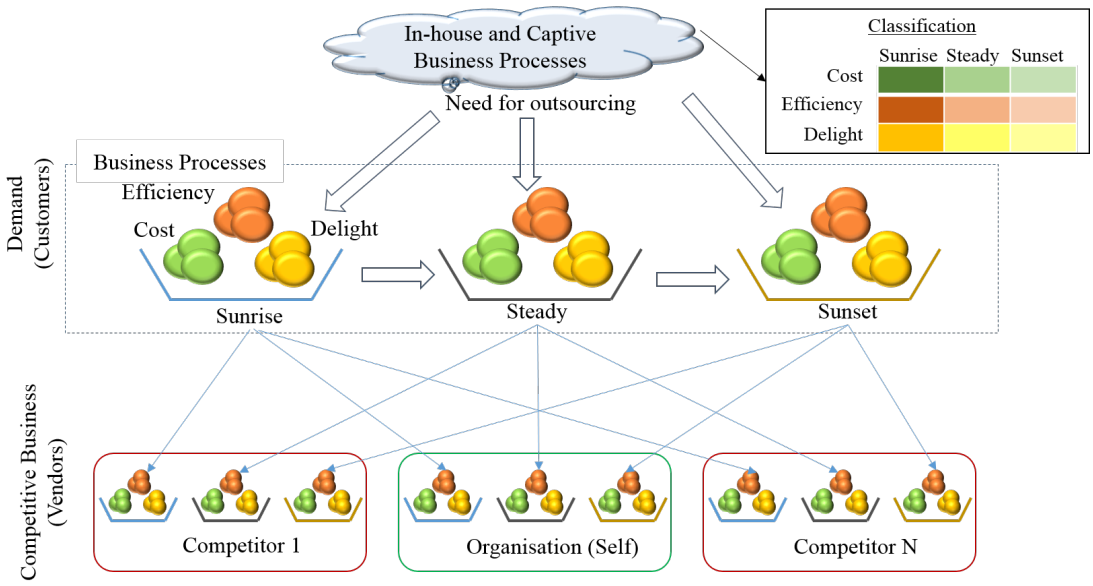


Figure E.1 A pictorial representation of Business Process Outsourcing organisation

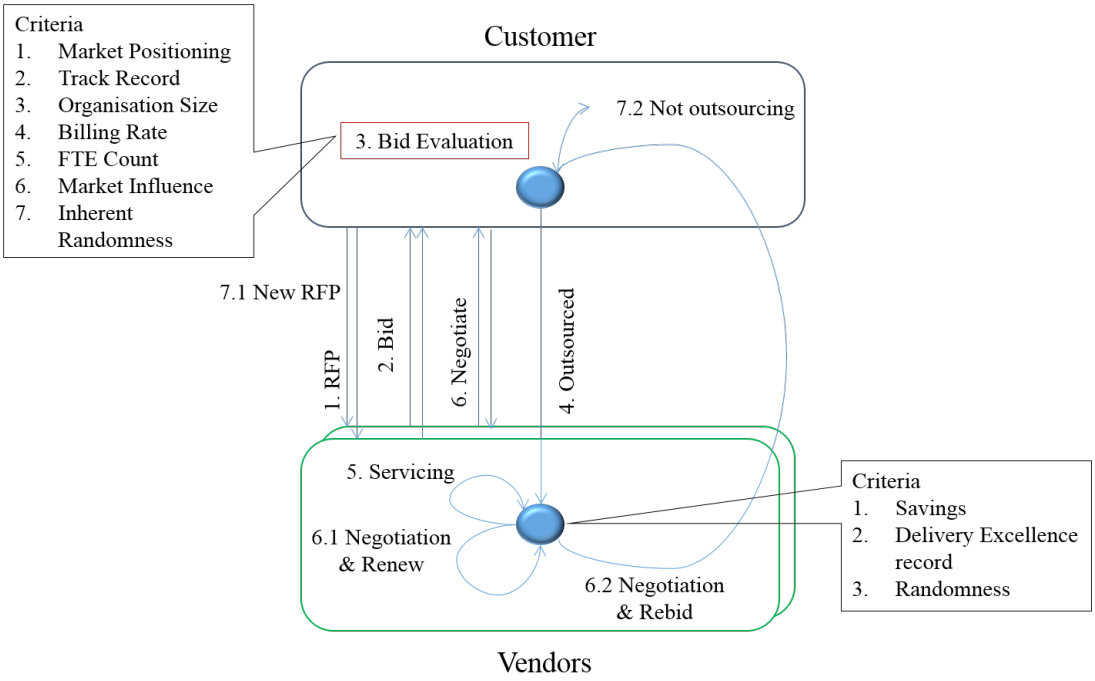


Figure E.2 Typical interactions and transitions in BPO environment

E.0.1 Problem entity

In BPO space, the organisations, termed as *customers*, outsource their business processes for a variety of reasons such as reducing cost (C), increasing efficiency (E), bringing about a major transformation or delightment (D). As shown in Figure E.1, the outsourced processes can be further classified into three buckets based on maturity of the BPO business. For instance, Transcript Entry process of Healthcare vertical was one of the first to take to BPO and has derived almost all potential benefits gained from outsourcing (*i.e.*, Sunset or SS). On the other hand, IT Infrastructure Management process being a late adopter of BPO has a large unrealized potential to be tapped (*i.e.*, Sunrise or SR). And there are processes such as Help Desk, Account Opening, Monthly Alerts *etc.*, that fall somewhere in between the two extremes as regards benefits accrued from BPO (*i.e.*, Steady or ST). Thus, BPO demand space can be viewed as a 3 x 3 matrix as depicted in Figure E.1.

The *customer* of a BPO business invites bids from the *vendors* for a specific business process. Typically, the factors such as quadrant (*i.e.*, ranking as per independent agency such as analysts), FTE count range (*i.e.*, min-max count of full time employee (FTE) to be deployed on the outsourced process), billing rate range (*i.e.*, min-max range for per hour rate of FTE), market influence (*i.e.*, perception of the market as regards delivery certainty with acceptable quality)

etc. decide who wins the bid. Other soft issues such as familiarity with the processes being outsourced, rapport with the vendor *etc.*, also play a part in selection of the vendor. It is common observation that BPO contracts come up for renewal after few years. Customer may renew the contract with the existing vendor on modified terms (typically advantageous to the customer) or may opt for rebidding. Factors influencing the renewal decision are reduction offered in FTE count, billing rate, number of escalations during service period, *etc*. Contracts that fail to get renewed become candidates for an open bidding. Figure E.2 shows the key interactions and transitions between *customers* and *vendors*.

The demand exhibits temporal dynamism and stochasticity. For instance, new processes emerge as candidates for outsourcing and some of the existing processes no longer need to be outsourced as, say, technology advance eliminates the need for human intervention in the process thus making it straight-through. While operating in this uncertain space, a BPO vendor needs to make decisions of the following kind: Will continuation with the current strategy (*e.g.*, with current FTE count, billing rate, market influence) keep a vendor viable for next 'n' years? What alternative strategies are available? How effective will a given strategy be (*e.g.*, different FTE count and billing rate)? By when a given strategy will start showing positive impact? and so on.

Answers to the above questions are primarily linked to the evaluation of portfolio basket, *i.e.*, 3 x 3 matrix of Figure E.1, of the organisation in terms of revenue and expenses. Therefore, ability to predict portfolio basket of the organisation and its competitors after a given time period becomes critical to support informed organisational decision making. The rest of this section explores two decision questions as follows:

1. Will continuation with the current strategy keep a vendor viable 'n' years hence with respect to its competitors?

2. What will be the situation if the vendor change required FTE count and billing rate?

E.0.2 OrgML model

The BPO problem entity is modelled as a set of autonomous and interacting `OrgUnits`. As shown in Figure E.3, the customers, vendors, business processes and the resources of the vendors are modelled as *Customer*, *Vendor*, *BusinessProcess*, *Resource* `OrgUnits` respectively. The interactions shown in Figure E.2 are realised as `IncomingEvents` and `OutgoingEvents` of

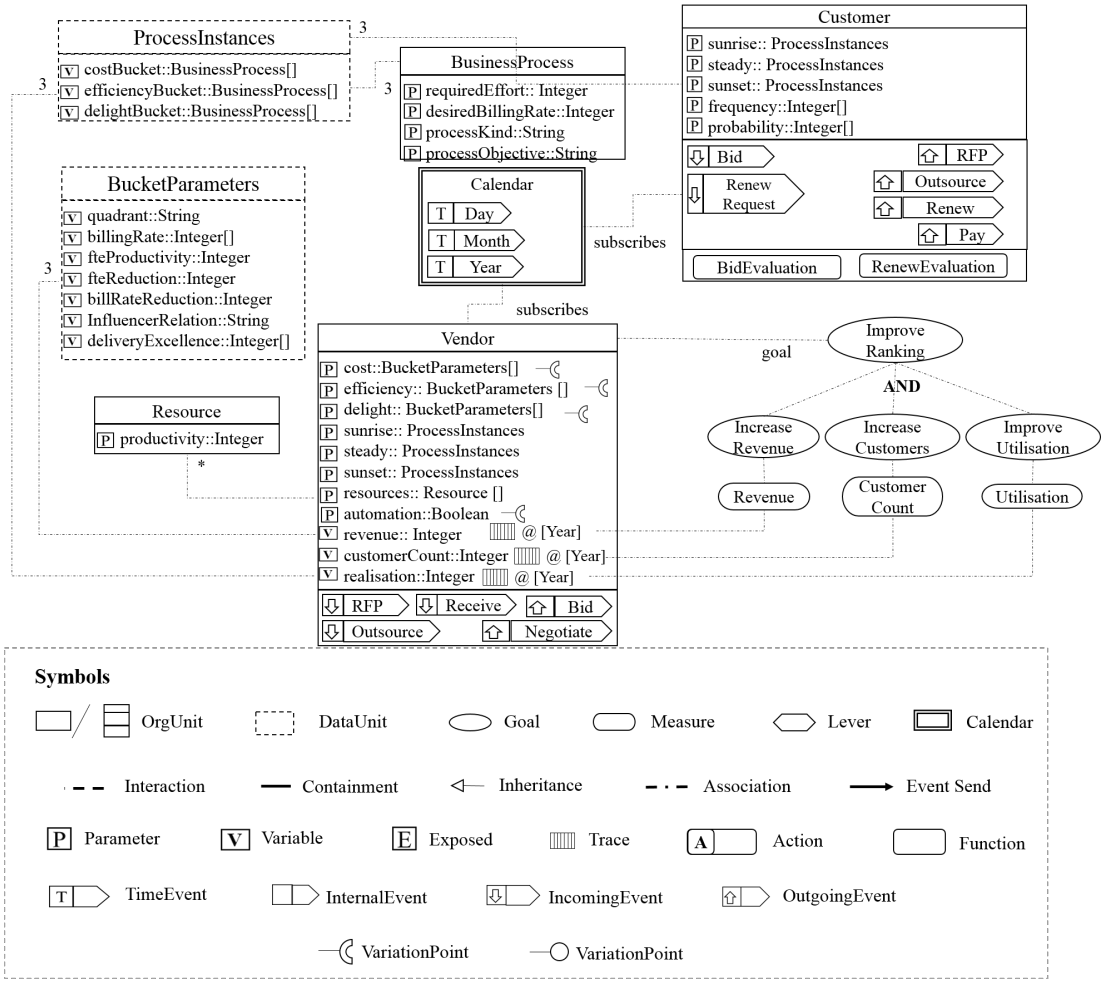


Figure E.3 OrgML specification of Business Process Outsourcing organisation

Customer and *Vendor* `OrgUnits`. The classification shown in Figure E.1 are specified using parametric `Variables` of *BusinessProcess* `OrgUnit`. The *Customer* and *Vendor* `OrgUnits` subscribe to a primitive `TimeEvent` termed as '*Day*' and two composite `TimeEvents`: '*Month*' and '*Year*'. The key elements of OrgML model are described below:

- **Customer**: *Customer* `OrgUnit` comprises three buckets for sunrise (SR), steady (ST) and sunset (SS) business processes where each bucket comprises a set of cost (C), efficiency (E) and delight (D) kinds of business processes. These buckets of buckets of business processes, *i.e.*, the bucket to represent 3 x 3 matrix of Figure E.1, are specified using '*sunrise*', '*steady*', '*sunset*' parametric `Variables` of '*Customer*' `OrgUnit` and a `DataUnit` termed as '*ProcessInstances*'. The increase and decrease in demand are specified using pre-defined frequencies and probabilities.

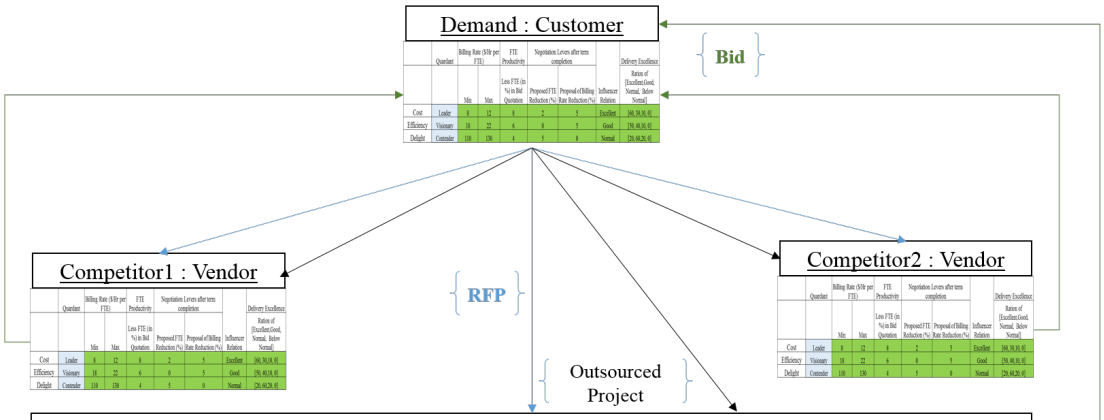
A *Customer* `OrgUnit` raises '*RFP*' events based on specified frequency and probability. Each '*RFP*' event is characterized by the kind of process being outsourced (*i.e.*, SR or ST or SS), the objective for outsourcing (*i.e.*, C or E or D), required effort (in terms of FTE count) to execute business process, and the expected billing rate.

As response to a '*RFP*', the *Customer* receives multiple '*Bids*' from *Vendors* and evaluates those Bids using '*BidEvaluation*' `Function`. The '*BidEvaluation*' function is a weighted aggregate of the various elements of RFP response and a random value to capture effect of inherent uncertainty as shown in Figure E.2. The vendor with the lowest bid wins the outsourcing deal which is communicated to specific vendor through '*Outsource*' event. *Customers* '*Pay*' every '*Month*' for their outsources business processes based on agreed billing rate and number of FTE. A *Customer* receives '*RenewRequest*', evaluates renew request using '*RenewEvaluation*' function and communicates decision through '*Renew*' event.

- **Vendor**: *Vendors* are typically characterised by a set of parameters such as its quadrant, typical billing rate (a range of min and max value), a trend of FTE count, market influence and delivery excellence, *etc*. The *Vendor* `OrgUnit` captures these values using '*cost*', '*efficiency*', '*delight*' parametric `Variables` and associated '*BucketParameter*' `DataUnit`. The resources of a *Vendor* is captured using '*resource*' parameter and *Resource* `OrgUnit`. The portfolio of the *Vendor*, *i.e.*, buckets of nine kinds of business processes (each from 3 x 3 matrix of Figure E.1), are captured using '*sunrise*', '*steady*', '*sunset*' parametric `Variables` and '*ProcessInstances*' `DataUnit`. In addition, *Vendors* `OrgUnit` captures revenue, customer counts, realisation (*i.e.* revenue per hour per resources), their yearly traces as shown in Figure E.3.

  Vendors create a competitive environment in a BPO space as all *Vendors* aim to improve their ranking with respect to their revenue, customer base and utilisation from their competitors. The `Goal`, goal decomposition, and `Measures` are captured using GM–L structure of *Vendor* `OrgUnit` as illustrated in Figure E.3.

  As shown in the figure, the parameters such as quadrant, billing rate, FTE count, market influence and delivery excellence are considered as `Levers` as they may help a *Vendor* to win a bid.



	Quardant	Billing Rate ($/Hr per FTE)		FTE Productivity	Negotiation Levers after term completion		Influencer Relation	Delivery Excellence
		Min	Max	Less FTE (in %) in Bid Quotation	FTE Reduction (%)	Billing Rate Reduction (%)		Ration of [Excellent,Good, Normal, Below Normal]
Cost	Leader	8	12	8	2	5	Excellent	[60, 30,10, 0]
Efficiency	Visionary	18	22	6	0	5	Good	[50, 40,10, 0]
Delight	Contender	110	130	4	5	0	Normal	[20, 60,20, 0]

Figure E.4 Input parameters for Business Process Outsourcing case study

- **BucketParameters**: BucketParameter is a `DataUnit` that holds a set of parametric `Variables` to characterise *Vendor* `OrgUnit`. Variable '*quadrant*' specify the ranking in a magic quadrant as per independent agency such as Gartner[1]. In BPO space, the quadrants are typically named as 'Leader', 'Visionary', 'Contender' and 'Niche player'. Variables billing rate and FTE productivity both are ranges and a value is picked at random from the specified range. The delivery excellence variable is a probability distribution of delivering 'Excellent', 'Good', 'Normal' and 'Below Normal' quality for a kind of BPO engagement. *BucketParameter* `DataUnit` contains two negotiation parameters: 'fteReduction' (*i.e.* what is the percent reduction possible in number of FTE billed against the outsourced process) and 'billRateReduction' (*i.e.*, what is the percent reduction possible in per hour billing rate for FTE).

- **Calendar**: A `Calendar` element is configured by specifying three `TimeEvents` that represent '*Day*', '*Month*' and '*Year*' where '*Day*' is associated with primitive `TimeEvents` and rest are specified as composite `TimeEvents`.

[1] https://www.gartner.com/doc/3650017/magic-quadrant-customer-management-contact

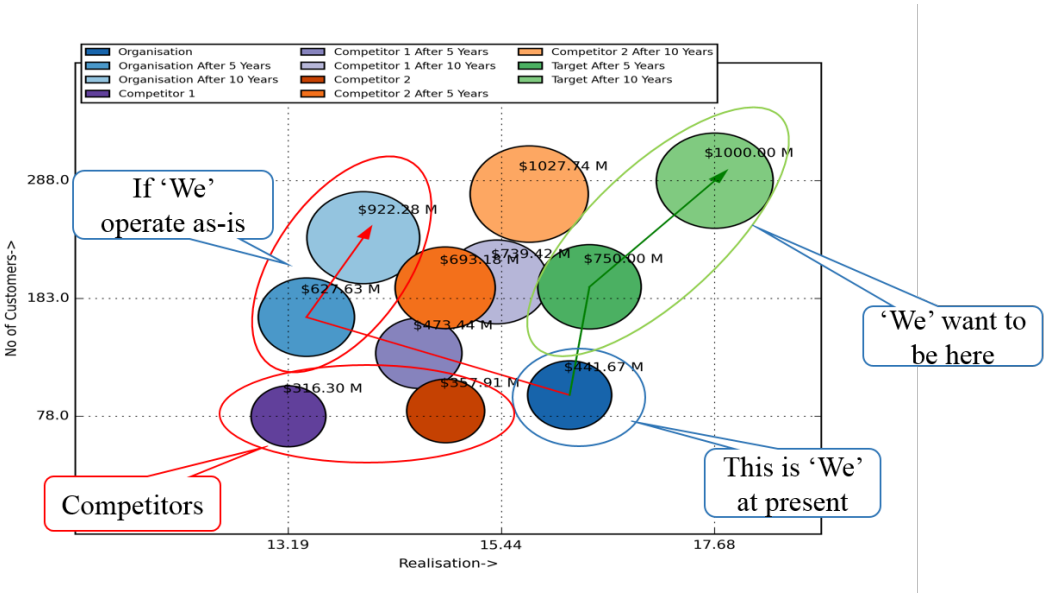


Figure E.5 Simulation dashboard of Business Process Outsourcing case study

E.0.3 Instantiation, simulation and decision making

An instance of BPO space is created with one *Customer* `OrgUnit` instance having a pool of various kinds of business processes, a *Vendor* termed as '*We*' and two competitor *Vendors*: '*Competitor1*' and '*Competitor2*' as shown in Figure E.4. The parameters of the '*We*' and competitor *Vendors* are appropriately set to create a competitive BPO environment. As shown in the figure, the '*We*' vendor is best equipped to win BPO contracts aimed at cost reduction. The vendor '*We*' is positioned in 'Leader' quadrant. It charges 8–12 USD per hour, offers around 8% less FTE with respect to standard FTE deployment and has'Excellent' relationship with influencer. The '*We*' vendor is confident of delivering 'Excellent' quality on 60% of 'cost' kind of BPO projects won. Similarly, the values for 'Good', 'Normal' and 'Below Normal' quality for 'cost' kind of BPO projects are respectively 30%, 10% and 0%. At the time of renew negotiation, the '*We*' vendor is equipped to offer 2% FTE reduction and 5% billing rate reduction. The '*Competitor1*' and '*Competitor2*' are also instantiated on the same lines as '*We*' vendor. In this case study two competitors are instantiated for simplicity. However, multiple competitors with different characteristics can be instantiated to define a complex BPO environment.

The above configuration is manually translated into ESL specification by applying the transformation rules defined in Chapter 5 and allowed to run for 10 '*Years* using ESL simulation engine. Results of the simulation run produced by OrgViz Data Visualiser are shown in Figure E.5. As can be seen, the current revenue of '*We*' vendor is 446.54 MUSD from 90

Vendor		Quardant	Billing Rate ($/Hr per FTE)		FTE Productivity	Negotiation Levers after term completion		Influencer Relation	Delivery Excellence
			Min	Max	Less FTE (in %) in Bid Quotation	FTE Reduction (%)	Billing Rate Reduction (%)		Ration of [Excellent,Good, Normal, Below Normal]
cost	Cost	Leader	12	14	10	5	1	Excellent	[90, 10,10, 0]
efficiency	Efficiency	Visionary	25	30	10	5	1	Good	[80, 10,10, 0]
delight	Delight	Contender	110	120	10	5	1	Excellent	[70, 30,0, 0]
…									

(a) Lever Specification

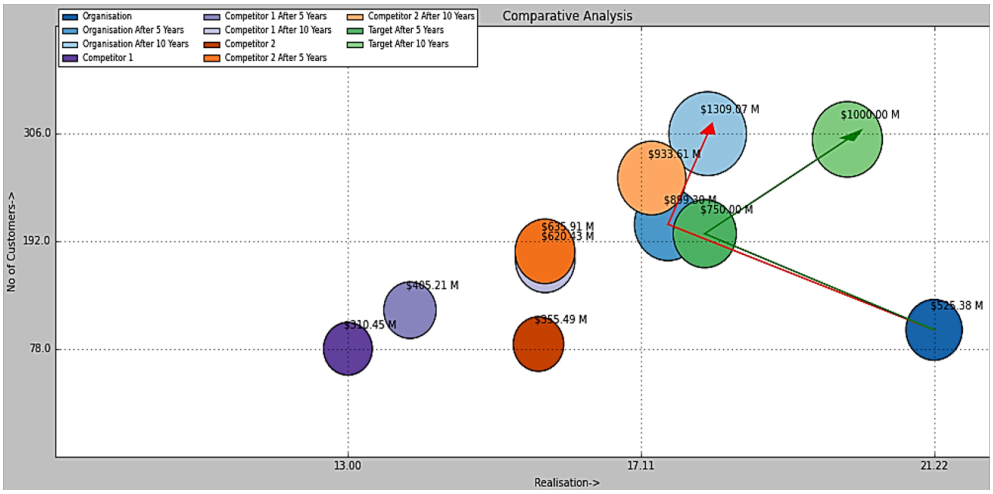


Figure E.6 Quantitative comparison

customers with a realization of nearly 17 USD per hour per FTE. Corresponding numbers for '*Competitor1*' and '*Competitor2*' vendors respectively are: <307.11, 78, 12.74> and <362.14, 80, 15>. In short, at present '*We*' vendor is doing much better than competition.

'*We*' vendor sets a goal to deliver <750, 200, 17> after 5 years and <1000, 290, 18> after 10 years. As can be seen, by continuing to operate the same way the '*We*' vendor will be delivering <621.81, 160, 13.5> after 5 years and <895.6, 215, 14> after 10 years thus missing both the targets by a considerable margin. More importantly, '*Competitor2*' vendor will be overtaking '*We*' vendor after 5 years and both the competitors will be significantly ahead of '*We*' vendor after 10 years.

Clearly, the '*We*' vendor cannot afford to continue with the current way of operation. Therefore, '*We*' vendor needs to bring about a change in its characteristics so as to be able to win more bids in this environment. Figure E.6 (a) shows a `Lever` that modifies the characteristics of '*We*' vendor. The improved performance of '*We*' vendor after applying `Lever` is shown in Figure E.6 (b). As shown in the figure, the '*We*' vendor is able to beat both revenue and customer targets while failing to meet the realization target narrowly.

E.0.4 Summary

This case study models a competitive environment where a set of vendors (with same or similar objective) compete to achieve their goals. The bid evaluation and renewal of an outsourcing engagement are specified as functions over track record of the participating vendors (*i.e.*, `Traces`, which changes over time), adopted strategies (*i.e.*, offered billing rates and FTE counts) and an inherent uncertainty. An ability to specify such realistic scenarios (as opposed to a fixed winning rate as a probability distribution) shows an advancement over the state-of-the-practice modelling and analysis techniques, such as spreadsheet, algebraic equations and Stock-and-Flow.

The quantitative evaluation of the *what-if* scenarios demonstrate the effectiveness of the proposed OrgML based approach for the organisations, which are competing with each other to achieve their goals in an uncertain, complex and nonlinear environment.

Appendix F

Multi-modelling and co-simulation using Enterprise Modelling techniques

Systematic literature review on Enterprise Modelling (EM) techniques presented in Chapter 4 identifies several inadequacies of EM techniques to use them as an aid for organisational decision-making. Reviews show that the existing EM techniques that are capable of specifying the necessary organisational aspects for organisational decision-making, such as ArchiMate [100], lack support for required analyses, whereas the EM techniques that are amenable for analysis can cater to specify only a subset of the aspects. For example, i* [218], BPMN [209], ARIS [175] and Stock and Flow (SnF) [134] are machine interpretable specification and they are amenable for a range of analyses. However, BPMN and ARIS are suitable for organisational processes, i* is limited to analyse organisational goals and objectives, and SnF focuses on business dynamics of the organisation. These observations leads to evaluate the efficacy of the *multi-modelling and co-simulation* approach to address the analysis needs for organisational decision-making.

This chapter presents an experiment on multi-modelling and co-simulation that combines i*, BPMN and SnF for *what-if* analysis of a Software Service Provisioning Organisation (SSPO) (which is presented in section 7.1 as a case study). The rest of this chapter is organised as follows – a brief description of SSPO and its goals is presented in section F.1, the experimental setup and adopted methodology are discussed in section F.2, *what-if* analyses of SSPO for a decision-making scenario are highlighted in section F.3, and finally the chapter concludes with a synthesis derived from this experiment in section F.4.

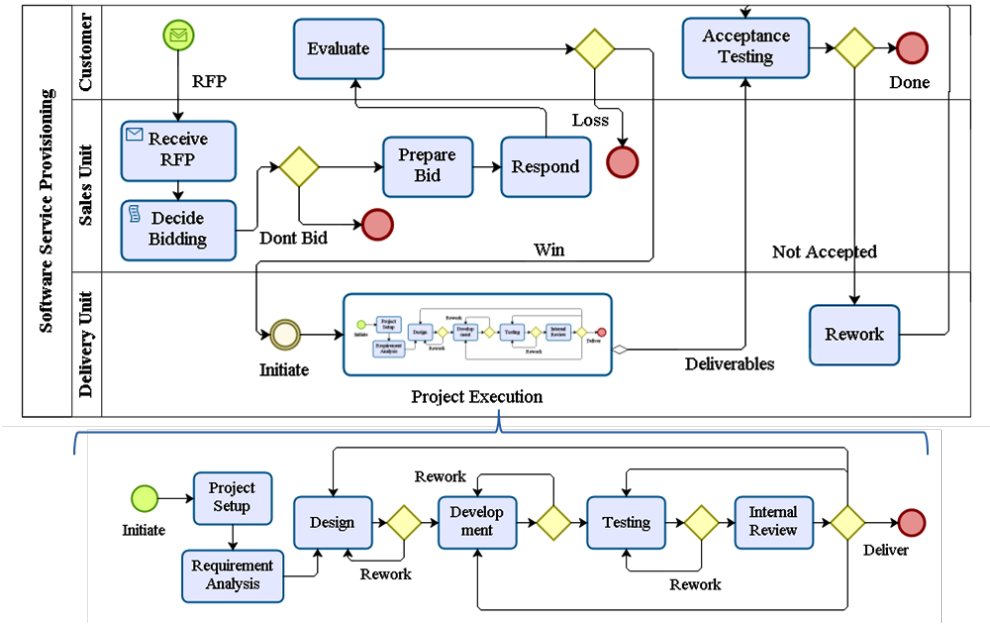


Figure F.1 Business process for software provisioning

F.1 Software service provisioning organisation

Consistent with the case study presented in section 7.1, this experiment considers a SSPO aims to secure leadership position in terms of business volume, profitability and customer satisfaction. In order to achieve these goals, SSPO adopts an operational process as shown in Figure F.1.

The organisation explores several strategies or levers to maximise its goals. Some strategies focus on introducing local fixes through improving operational efficiency while keeping structural as well as process aspects of the organisation unchanged. For instance, one can think of increasing number and skill-level of the existing resources, reducing resource attrition, training *etc*. Some strategies might be more disruptive as they introduce changes in the organisation structure and/or operational processes. For example, one can think of developing *productivity improvement* tools, which necessitates a change in project execution process as well in the skill-set of project team.

This experiment evaluates some of these strategies using i*, BPMN and SnF. The i* model specifies organisational goals, BPMN model specifies operational processes, SnF specifies aggregated business dynamics of the SSPO. Precise experimental setup is discussed in the next section.

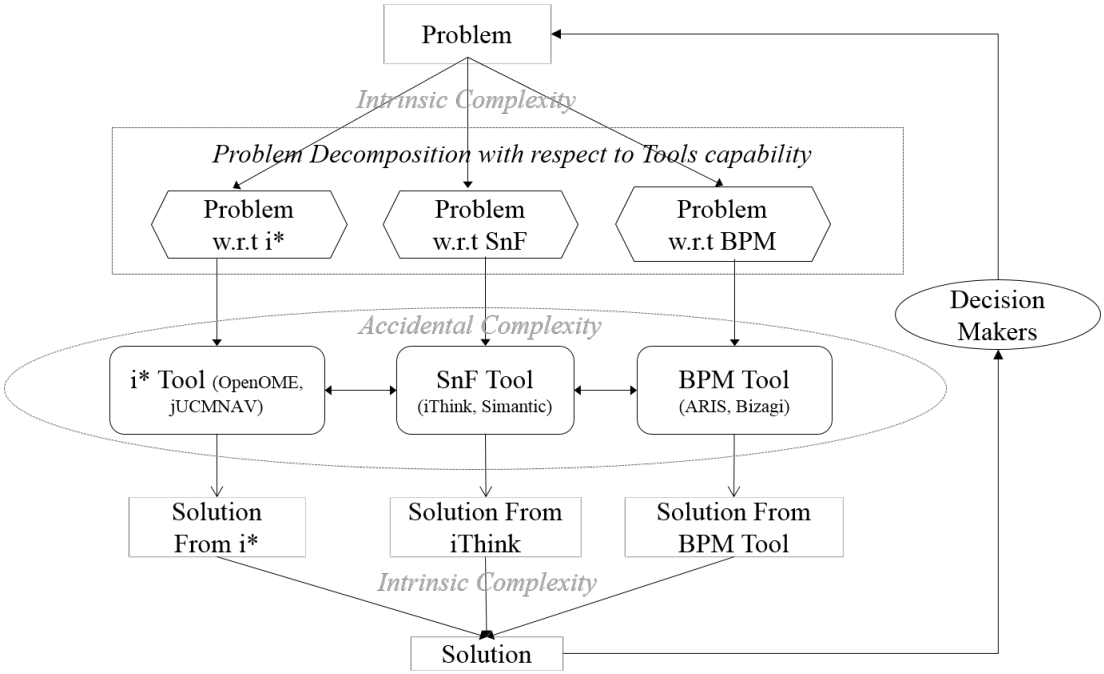


Figure F.2 Multi-modelling and co-simulation in organisational decision making

F.2 Environment for multi-modelling and co-simulation

The capabilities of a wide spectrum of enterprise modelling, analysis and simulation techniques to represent and analyse complex systems and enterprises are discussed in Chapter 4. Table 4.2 highlights prominent EM techniques and Table 4.3 shows their capabilities with respect to the modelling and analysis needs for an effective organistional decision-making. This experiment chooses i*, BPMN and SnF as collectively they are capable of representing the required organisational aspects as shown in Table 4.3. Moreover, i* tools, *e.g.* OpenOME[1], support qualitative and quantitative analysis of organisational goals, BPMN tools (*e.g.* Bizagi[2]) are capable of quantitative analysis and simulation of business processes, and SnF tools, such as iThink[3] and Simantics[4], come with a rich simulation machinery supporting *what-if* simulation.

This experiment adopts a reductionist view to visualises a decision-making problem into multiple sub-problems (*i.e.*, multiple *what-if* scenarios) as these tools can model and analyse only a partial view of an organisation. In this experiment, the required *what-if* analyses are divided into three categories such that each category of *what-if* analysis can be addressed using

[1] www.cs.toronto.edu/km/openome
[2] https://www.bizagi.com
[3] http://www.iseesystems.com/Softwares/Business/ithinkSoftware.aspx
[4] www.simantics.org

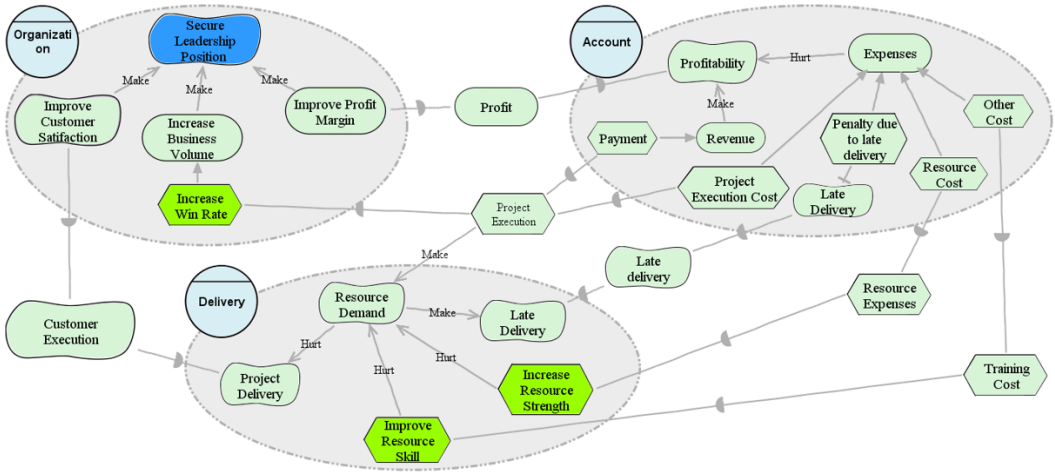


Figure F.3 Elaborated i* model

i*, BPMN or SnF as shown in Figure F.2. Precise categorisation, analyses and decision making by judiciously integrating the partial solutions obtainable from various tools are discuss in the next section.

F.3 Multi-modelling, co-simulation and decision making

The decision-making for SSPO starts with an i* model having a root goal '*Secure Leadership Position*'. The root goal is then elaborated into sub-goals and their decompositions with several alternate levers made explicit. Figure F.3 shows elaborated i* model where '*Improve Customer Satisfaction*', '*Increase Business Volume*' and '*Improve Profit Margin*' constitute first-level elaboration of the root goal '*Secure Leadership Position*' (which is marked with blue colour). Lever '*Increase Win Rate*' is identified as a means for realising elaborated goal '*Increase Business Volume*'. '*Increase Customer Satisfaction*' sub-goal is dependent on *Softgoal* '*Project Delivery*', which is further influenced by a *Softgoal* '*Resource Demand*' where '*Resource Demand*' could be managed by two levers: '*Increase Resource Strength*' and '*Increase Resource Skill*'.

The goal '*Improve Profit Margin*' is dependent on *Softgoal* '*Profitability*', which is then refined into two sub-goals: '*Revenue*' and '*Expenses*' along with identification of possible levers for achieving the two. The model depicted in Figure F.3 is essentially a subset of GM-L structure where goal, goal decomposition structure and levers are modelled using i* model.

Table F.1 Qualitative Analysis using i* model

	Levers	**Goal and Sub-goals** (symbols :- ✓ - Satisfied, ✓ Partially Satisfied , ✗ Partially Denied, ↘ Conflict)						**Principal Goal**
		Increase Business Volume	*Revenue*	*Expense*	*Profitability*	*Late Delivery*	*Improve Customer Satisfaction*	*Secure Leadership Position*
1	Increase Win Rate	✓	✓	✓	↘ SnF1	✓	✗	↘
2	Increase Resource Strength	-	-	✓	✗	✗	✓	↘
3	Improve Resource Skill	-	-	✓	✗	✗	✓	↘
4	L1 + L2	✓	✓	✓	↘ SnF2	↘ BP	↘	↘

Iterative analyses of constructed i* model provides a qualitative insight into the possible impact of levers on various sub-goals that eventually percolate to the root goal. The *decision table* shown in Table F.1 depicts the impact of three levers (*i.e.*, '*Increase Win Rate*', '*Increase Resource Strength*' and '*Increase Resource Skill*') on selected sub-goals and goals. Table also depicts the analysis results for applying levers '*Increase Win Rate*' and '*Increase Resource Strength*' together. For instance, lever '*Increase Win Rate*' will: i) positively impact '*Improve Business Volume*', '*Revenue*', '*Expense*' and '*Late Delivery*' goals, ii) negatively impact '*Improve Customer Satisfaction*' goal, and iii) is inconclusive about '*Profitability*' goal. Thus, nothing conclusive can be said about the impact of this lever on the root goal. Table F.1 clearly identifies which decision points are left unaddressed (*i.e.*, decision-points marked as `SF1`, `SF2` and `BP`). Moreover, decision maker would like to have a quantitative feel for some of the qualitatively arrived decisions.

This constitutes the next step of decision-making. The next step uses either SnF or BPMN for understanding the impact of specific lever(s) on goals. The SnF tool iThink and BPMN tool Bizagi are used for next set of *what-if* explorations. Precisely, the decision points `SnF1` and `SnF2` of Table F.1 are addressed using SnF model as they require quantitative and temporal analysis on aggregated business operations to understand when overall '*Revenue*' may supersede the overall '*Expenses*'. On the other hand, the decision point `BP` is addressed using business process model as it requires simulation of operational processes to understand the percentage of (individual) projects that may get delayed due to delays in '*Project Setup*', multiple iterations due to '*Rework*' in '*Project Execution*' business process (*see* Figure F.1), *etc*.

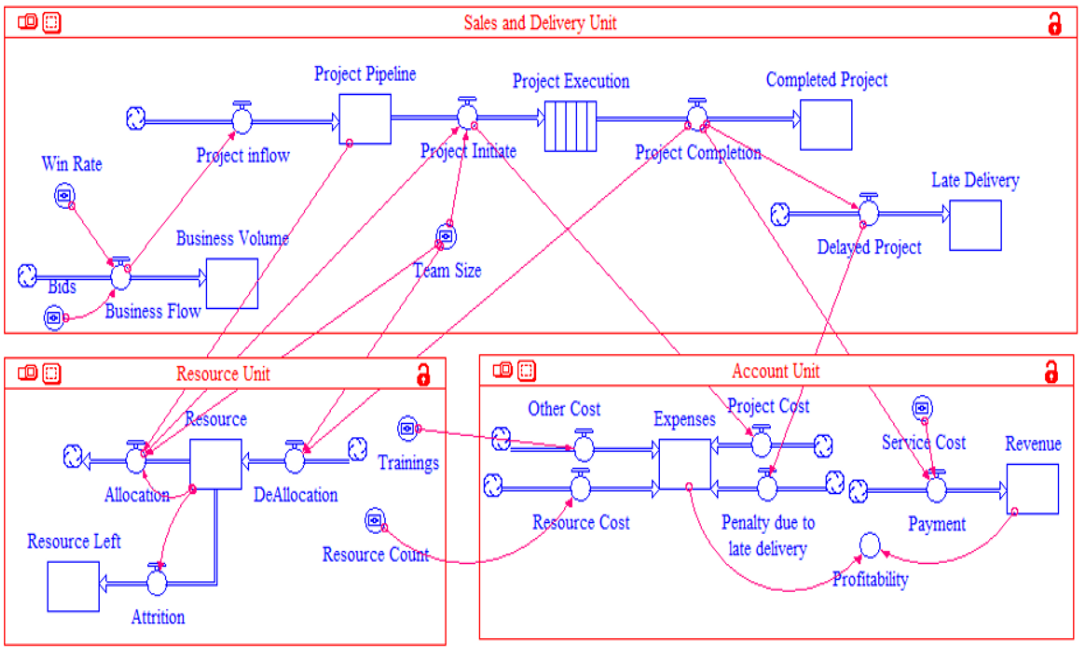


Figure F.4 Stock-and-Flow model of Software Service Provisioning Organisation

A necessary and sufficient SnF is constructed for *what-if* analyses formulated in `SnF1` and `SnF2` decision points as shown in Figure F.4 and the business process model depicted in Figure F.1 is used for `BP` decision point. Constructed SnF model focuses on the '*Profitability*' goal of i* model depicted in Figure F.3. The '*Profitability*' goal is represented using 'Profitability' *Auxiliary* variable within Account Unit of SnF model. The '*Revenue*' and '*Expenses*' goals are represented using 'Revenue' and 'Expenses' *Stocks*. The *Tasks* of i* model that contribute to '*Revenue*' and '*Expenses*' goals using means-ends links are represented using inflow *Flows*. For example, '*Payment*' is represented using 'Payment' Flow to 'Revenue' *Stock*. The rest of the model is created by navigating back to the dependent goals and levers. For example, the impact of 'Increase Win Rate' *Task* of i* model is represented using 'Win Rate' *Auxiliary* variable and subsequent *Stock*, *Flows* and *Connectors*; the path '*Increase Win Rate*' and '*Increase Business Volume*' are represented using 'Win Rate' *Auxiliary* variable, 'Business Flow' inflow and 'Business Volume' Stock. The 'Project Execution' *Task* of i* model is a complex activity and hence expanded further while constructing the SnF model. The expansion is illustrated using Stock-and-Flow path 'Project inflow' *Flow* to 'Completed Project' *Stock*. The project associated delays and the penalty due to late delivery are considered using 'Delayed Project' *Flow*, 'Late delivery' *Stock* and *connectors*.

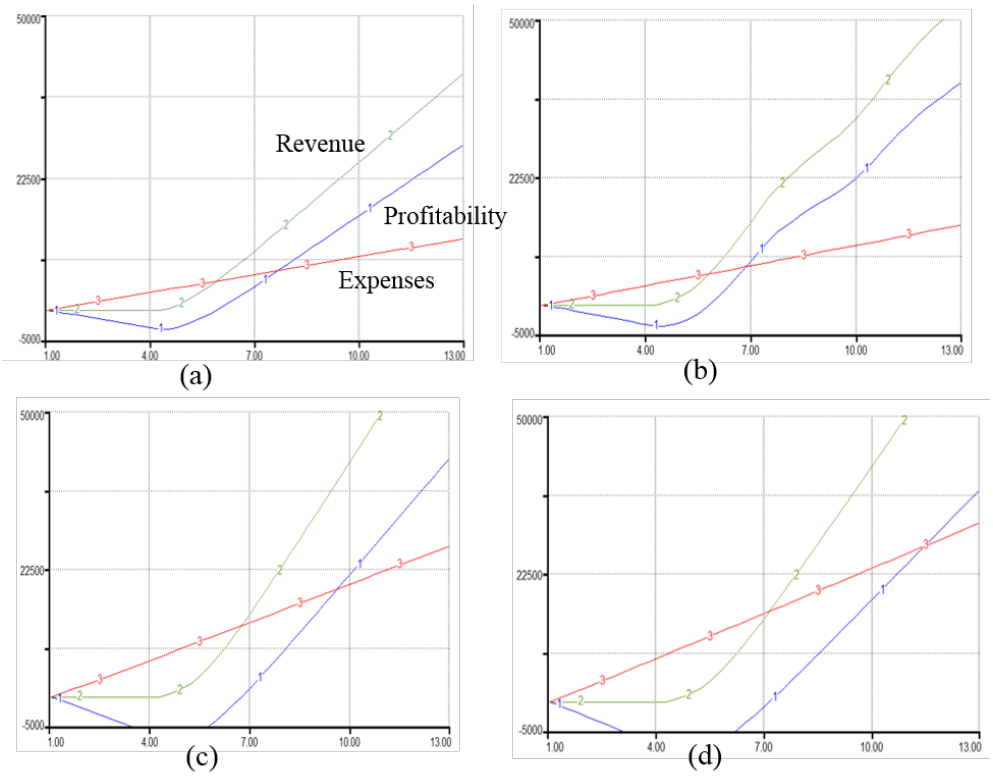


Figure F.5 Quantitative analysis using Stock-and-Flow model for profitability

Table F.2 Results of what-if analysis using simulation model

	Levers	**Goal, Sub-goals and other Outcomes** (symbols :- ↑ - increase, ↓ - decrease , ↕ - unknown, □ - eventually, Δ - delta)						**organization Goal**
		Business Volume	*Revenue*	*Expense*	*Profitability*	*Late Delivery*	*Customer Satisfaction*	*Secure Leadership Position*
1	Increase Win Rate	↑	(Δ)↑	(Δ)↑	(□Δ)↑	↑	(□)↓	↕
2	Increase Resource Strength	–	–	↑	↓	↓	↕	↕
3	Improve Resource Skill	–	–	(Δ)↑	(□Δ)↓	(Δ) ↓	(Δ)↑	↕
4	L 1 + L2	↑	↑	↑	↑	↓	↑	↑

Simulation results of the constructed SnF with suitable data are shown in Figure F.5 and summarisation of the simulation data is recorded in a decision table as shown in Table F.2. The quantitative and temporal analysis result of lever '*Increase Win Rate*' on goal '*Profitability*' (*i.e.*, SnF1) is shown using a graph in Figure F.5 (a) and the impact of levers '*Increase Win Rate*' and '*Increase Resource Strength*' together on '*Profitability*' goal is shown in Figure F.5 (b). As can be seen from Figure F.5, the profitability drops initially but improves over time leading to positive impact for both the alternatives. If unsatisfactory, one can keep on modifying value of

the Auxiliary variable 'Resource Count' to evaluate the impact of lever. 'Increase Resource Strength' in this combination - Figure F.5 (c) and Figure F.5 (d) depict such iterations. On the other hand the simulation of business process depicted in Figure F.1 provides an insight about BP decision point. Simulation result shows '*Late Delivery*' reduces to an extent with '*Increase Resource Strength*' with reduction in delays in 'Project Initiate' task and re-initiating tasks that traverse through 'Rework' loop. Therefore, the goal '*Improve Customer Satisfaction*' improves with the combination of L1 and L2 of Table F.2. As shown in the table, the lever L1 and L2 together help to achieve '*Secure Leadership Position*' goal of SSPO. There could be many such iterations over SnF and business process model simulations considering i* model as a navigation aid for exploring options to reach a satisfactory answer.

F.4 Synthesis

The above experiment decomposes the decision-making problem of Software Service Provisioning Organisation into three sub-problems such that they can be addressed using <i*, OpenOME>, <SnF, iThink>, and <BPMN, Bizagi>. First, the goal is qualitatively analysed using i* model and a set of decision points are identified where the precise quantitative analyses are useful. For each such decision point or a specific set of decision points, an appropriate and purposive model is constructed using a specific formalism, and then the *what-if* analyses are carried out for decision-making. For example, an SnF is constructed for decision point SF1 and SF2, and a BPMN model is constructed and analysed for decision point BP. Finally, these partial solutions, which are obtained from separate tools, are integrated into a consistent whole using decision table as shown in Table F.2 for organisational decision-making.

This experiment demonstrates that a judicious and systematic use of a set of EM techniques and tools can address a class of organisational decision-making problems where the organisation is largely mechanistic, organisational behaviour is precisely known and it can be represented using aggregated equations. However, this approach is prone to two kinds of complexities: *intrinsic complexity* and *accidental complexity*, as discussed in [120]. Two major factors that contribute to an *intrinsic complexity* are: (i) the need for decomposing an organisational decision-making problem into parts such that they can be addressed using existing tools, and (ii) an integration of the partial solutions obtained from disparate tools into a consistent whole for sense making. The overlapping specifications, inability to set up relationships across specifications,

and non-interoperable nature of the existing tools are principal contributors to the *accidental complexity*.

F.5 Summary

This experiment shows the benefits of multi-modelling and co-simulation approach over any of the individual EM technique. However, the inability to express individualistic behaviours and lack of analysis capability to understand emergentism of a complex system are remained an open question for an EM technique based multi-modelling and co-simulation approach as none of the EM technique is cognisant of such capabilities. Moreover, associated *intrinsic complexity* and *accidental complexity* make this approach difficult to use in organisational decision-making.